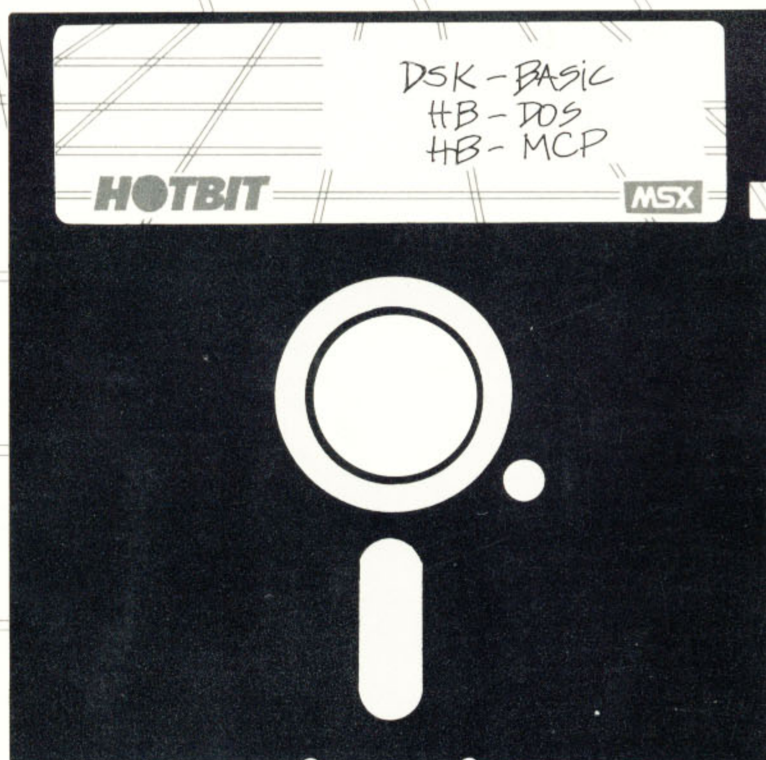
COLEÇÃO MSX



USANDO O DISK DRIVE NO MSX

MSX

USANDO O DISK DRIVE NO MSX





RUBENS PEREIRA SILVA JR.

USANDO O DISK DRIVE NO MSX

COLEÇÃO MSX

3ª EDIÇÃO



COORDENAÇÃO EDITORIAL: Pierluigi Piazzi
EDITORAÇÃO: Renato da Silva Oliveira
PRODUÇÃO EDITORIAL: Betty F. Piazzi
REVISÃO TÉCNICA: Eduardo Resende
ILUSTRAÇÃO E ARTE: Ana Lúcia Antico
CAPA: Alvaro Guillermo



ALEPH Publicações e
Assessoria Pedagógica Ltda.
Av. Dr. Luiz Migliano, 1110
Cj. 301/303 - Morumbi
05711 - São Paulo - SP
Fone: (011) 843-3202

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

S584u Silva Júnior, Rubens Pereira, 1961-
Usando disk-drive no MSX : DSK-BASIC, HB-DOS, HB-MCP / Rubens Pereira Silva Júnior. -- São Paulo : Aleph, 1986. (Coleção MSX)

1. CP/M (Sistema operacional de computador 2. MSX (Computador) - Programação 3. Programação (Computadores eletrônicos) 4. Sistemas operacionais (Computadores) I. Título. II. Série.

86-1009 CDD-001.642
-001.6425

Índices para catálogo sistemático:

1. Computadores : Programação : Processamento de dados 001.642
2. CP/M : Sistema operacional : Computadores : Processamento de dados 001.6425
3. MSX : Computadores : Programação : Processamento de dados 001.642
4. Programação de computadores : Processamento de dados 001.642
5. Sistemas operacionais em disco : Processamento de dados 001.6425

SUMÁRIO

APÊNDICES

NOTA DO EDITOR

Algumas semanas antes do lançamento deste livro, passei por Amsterdam para ver, entre outras coisas, o que os holandeses estavam fazendo, de interessante, com seus MSX.

Todos sabem o quão penoso é, para o viajante, se locomover numa cidade desconhecida onde se fala uma língua estranha. Em Amsterdam, porém, tive uma agradabilíssima surpresa! Toda cidade é servida por uma rede de BONDES (com horário e tudo!) tão fácil e racional de ser usada que em pouco tempo estava me deslocando com a desenvoltura de um nativo.

Em São Paulo, em compensação, havia uma linha de bonde que ligava o Instituto Biológico no Paraíso com Santo Amaro praticamente em linha reta. Este bonde tinha horários e era praticamente o embrião de uma rede de Metrô ao ar livre.

Pois bem, em nome de um estranho conceito de progresso esta linha, e toda rede de transporte por bonde de São Paulo, foi desativada. Quilômetros e quilômetros de trilhos foram enterrados pelo asfalto sob a alegação de que os bondes atrapalhavam os carros!

Isto me fez meditar um pouco sobre a mania que o ser humano tem de procurar a novidade pela novidade. Na realidade tecnologias mais antigas, se utilizadas de maneira moderna, produzem resultados modernos e muitas vezes a custos espantosamente mais baixos que os das "tecnologias de ponta"!

O estabelecimento do padrão MSX no mundo inteiro me lembra um pouco a utilização racional do bonde. Pegou-se o mais consagrado microprocessador de 8 bits (o famoso Z80), o mais eficiente gerador de som, o melhor processador de vídeo, um poderosíssimo sistema operacional e se criou uma maravilha a preço espantosamente baixo.

Sua receita é extremamente simples: componentes já existentes e muita inteligência.

Nos Estados Unidos, Japão e Europa, um microcomputador pessoal pode custar tranquilamente mais de 2000 dólares. No Brasil,qualquer sistema que exija mais de 1000 dólares para sua instalação já é vedado à quase totalidade do mercado doméstico e a uma parcela considerável do mercado empresarial, constituida por pequenas firmas, consultórios, escritórios de advocacia, etc.

Um MSX como o HOTBIT, utilizando cartuchos e um sistema de gravação de dados em fita, representa um sistema altamente utilizável para educação, lazer e uma série de aplicações domésticas importantes.

Para aplicações mais profissionais, onde a rapidez e a confiabilidade do armazenamento dos dados se tornam fatores cruciais, temos o disk drive, um importante periférico que permite a incorporação de outros poderosos sistemas operacionais além do residente.

Com o disk drive o usuário passa a dispor do DSK BASIC, uma extensão do BASIC residente que permite gerenciar os arquivos e programas gravados em disquete.

Além disso, a partir do próprio disco podemos carregar o HB-DOS, um sistema operacional extremamente versátil que permite a utilização de uma série de interpretadores e compiladores para outras linguagens. Traduzindo isto para uma linguagem simples, poderíamos dizer que este sistema permite transformar o HOTBIT e seus compatíveis MSX em poliglotas que além de falar sua própria língua (o BASIC) passam a utilizar as mais modernas linguagens de programação.

O uso do disk drive no HOTBIT e nos seus compatíveis permite a utilização do HB-MCP, um sistema operacional que compatibiliza o MSX com uma vastíssima biblioteca de software já desenvolvida para outros computadores profissionais.

Resumindo, podemos dizer que a utilização do disk drive nos computadores do padrão MSX os transforma em equipamentos moderníssimos, extremamente eficientes, assistidos por uma enorme quantidade de software já desenvolvido e, fator importantíssimo para a realidade econômica brasileira, de custo muito baixo.

O aparecimento do disk drive para o HOTBIT foi, com certeza, o mais importante passo dado nos últimos tempos para a verdadeira informatização de nossa sociedade.

Prof Pierluigi Piazzi
Editor

CAPÍTULO I



INTRODUÇÃO

De uma maneira geral, o trabalho com o computador resume-se em receber informações, tratar essas informações através de um algoritmo já conhecido (que é o programa que está carregado no computador) e devolver outras informações como resultado, sendo estas muito mais úteis do que as do início do processo.

Para que o computador possa receber informações, é necessário que haja um meio delas chegarem até ele. Por isso existem periféricos como o teclado, o gravador cassete e o disk-drive, entre outros.

O disk-drive (que daqui para frente vamos chamar apenas de drive) é um periférico que permite ao computador ler e gravar informações em disquetes (chamados também de discos, apenas), numa grande velocidade e com muita confiabilidade.

Quando utilizamos um drive, existem certas regras para que o computador possa receber e enviar informações a ele. Sem essas regras, ficaria muito difícil, senão impossível, ao drive se tornar útil.

Essas regras nada mais são do que extensões naquelas que você usa quando utiliza o computador com um gravador casset. Não será difícil, portanto, aprender como usar o drive se você souber como utilizar o gravador cassete.

Os dados gravados em discos podem ser acessados de uma maneira muito mais rápida e mais dinâmica que os dados armazenados com um gravador.

Para podermos entender e dominar todas as facilidades oferecidas pelo uso do drive, vamos primeiro nos familiarizar com alguns termos comuns na área de computação, para depois entrarmos com o pé direito naquilo que o HOTBIT oferece quando conectado ao drive: o DSK-BASIC, o HB-DOS e o HB-MCP.

As informações a serem armazenadas não podem simplesmente ser "jogadas" no disco. É preciso uma certa organização.

Toda pessoa de bom juízo sabe que em uma agenda de telefones encontraremos os telefones dos amigos e não receitas de bolos. Se quisermos saber como preparar uma torta de maçã, devemos procurar a receita em um livro de culinária e não numa lista telefônica! Dessa mesma maneira os dados a serem gravados no disco devem ser agrupados: as receitas, no livro de culinária, e os telefones, nas agendas.

ARQUIVOS

Vamos chamar de arquivo a um conjunto de informações que estão agrupadas de uma dada forma.

Por exemplo: a lista dos nomes dos alunos de uma sala de aula pode ser um arquivo. A lista telefônica pode ser outro arquivo. Vamos, portanto, chamar de arquivo a todo grupo de dados que estiverem relacionados entre si, formando um

conjunto.

Todas as informações gravadas em disco são obrigatoriamente agrupadas em arquivos.

Você pode imaginar o disco como sendo uma pequena biblioteca onde estão vários livros (que são os arquivos) e que cada livro contém informações sobre um dado assunto: culinária, telefones, esportes, etc.

REGISTROS

Dentro de um arquivo, portanto, nós temos muitas informações que podem e devem ser separadas em grupos menores que vamos chamar de registros.

Imagine uma lista de chamada, daquelas usadas pelos professores, como sendo um arquivo. Nela temos informações sobre o aproveitamento de cada aluno.

Imagine também uma lista telefônica como sendo outro arquivo.

No exemplo da lista dos nomes dos alunos em uma sala de aula,podemos chamar de registro ao seguinte conjunto: nome do aluno, número de faltas no mês, nota do aluno.

No exemplo da lista telefônica, nós podemos ter como registro o conjunto:nome do assinante e número do telefone.

A idéia de registro dentro de um arquivo de disco é a mesma idéia de uma ficha dentro de uma gaveta em um arquivo de escritório, ou seja: o arquivo é a união de todos os registros que contém um grupo de informações sobre cada item em separado.

CAMPOS

Dentro de um registro temos ainda várias informações sobre um dado item e convém separarmos esses dados mais uma vez. Vamos chamar a cada informação assim separada de campo.

Campo é a menor informação completa que existe dentro de um registro e, consequentemente, dentro de um arquivo.

Nos exemplos citados anteriormente, um campo seria o nome do aluno; outro campo seria o número de faltas daquele aluno no mês; outro seria o nome do assinante (no arquivo da lista telefônica) e assim por diante.

Para entender melhor as idéias de arquivo, registro e campo, vamos imaginar uma pequena agenda de telefones, dessas que geralmente carregamos conosco no bolso.

Nessa agenda (que é um arquivo) nós temos os nomes dos nossos amigos e clientes. Associados a esses nomes nós temos: endereço completo e telefone (que formam, juntos, um registro).

Os nomes, endereços e telefones são,cada um, um campo, pois representam informações completas elementares.

Veja na figura 1.1 essa nossa agenda desenhada de uma maneira bem simples para tirar qualquer dúvida que ainda exista.

FIGURA 1.1 - Agenda telefônica.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1º
r   NOME:________________________________________◄──campo "NOME"
e
g   END:_________________________________________◄──campo "END"
i
s   TEL:______________________  ◄─────────────────── campo "TEL"
t
r
o
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
2º
r   NOME:________________________________________
e
g   END:_________________________________________
i
s   TEL:______________________
t
r
o
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                              •
                              •
                              •
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
nº
r   NOME:________________________________________
e
g   END:_________________________________________
i
s   TEL:______________________
t
r
o
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

PROGRAMAS

Existe um tipo especial de arquivo que nós chamamos de PROGRAMA. Um programa é um conjunto de informações que foram agrupadas de forma a fazer com que o computador execute tarefas consecutivas e que tenham alguma utilidade prática. Os programas são gravados em disquetes como arquivos. A diferença entre um arquivo qualquer e um programa, é que este último contém informações para uso do computador enquanto os demais arquivos contém informações para nosso próprio uso, como nomes, telefones, etc. Se tentarmos olhar as informações contidas num programa, veremos apenas uma sequência aparentemente aleatória de caracteres (letras, números, símbolos gráficos, etc.), sem nenhum significado imediato para nós, porém perfeitamente compreensíveis para o computador. Na verdade, nós podemos usar os dados que constituem um programa, mas apenas através do computador.

Tanto os programas como os demais arquivos são tratados pelo computador com indiferença quanto a forma de serem lidos e gravados em disco. Para o computador, tudo não passa de um monte de dados, sendo que alguns ele "entende" (programas) e outros nós "entendemos" (demais arquivos).

NOMES DOS ARQUIVOS

CASSETE

Você deve ter notado que um arquivo necessita de um nome. Esse nome nos permite localizar e diferenciar esse arquivo dos vários outros que possam existir.

Em um computador, os arquivos podem estar gravados em fita cassete ou em disco e cada um desses arquivos precisa ter um nome. O nome de um arquivo tem certas restrições que precisam ser observadas.

Nomes de arquivos gravados em fita cassete não podem conter mais do que seis caracteres, ou seja, devem ter pelo menos uma letra e no máximo seis. Você pode dar o nome que bem entender aos seus arquivos, desde que ele não tenha mais do que seis caracteres e que cada caractere seja uma letra ou número.

Veja alguns nomes que podem ser usados como nomes de arquivos para fita cassete:

AGENDA	X1	K	TENIS
REDATO	ABK3	LIXO	QUADRO
ALUNOS	ARQTST	JOGO	DADOS1

DISQUETE

Os arquivos que forem gravados em disco têm uma flexibilidade muito maior a começar pelo nome. Você pode utilizar até oito caracteres para o nome do arquivo.

Quanto às restrições, continuam existindo: o nome deve ter pelo menos um caractere e no máximo oito, composto de letras e/ou números. Além desses oito caracteres, o nome de um arquivo em disco possui ainda mais três caracteres que servem como indicador de tipo, e que chamaremos de extensão do nome.

Portanto, onze é o número máximo de caracteres que compõem o nome do arquivo: oito para o nome propriamente dito e mais três para a extensão do nome (geralmente usado para identificar que tipo de dados estão gravados no arquivo).

Para separar os oito caracteres do nome do arquivo dos três caracteres da extensão do nome é utilizado um ponto (.).

O ponto que separa o nome da extensão não faz parte nem do nome e nem da extensão, ele serve apenas para separar um do outro.

Veja alguns exemplos válidos de nomes de arquivos em disco:

TESTE.BAS	JOGO.NEW	COMANDOS.BAT	A1.ABC
ALUNOS.B5	JANELA.COB	ARQALEPH.ASM	X.2

A extensão do nome do arquivo é opcional, você pode ou não colocá-la. Ela existe apenas para facilitar a nossa compreensão sobre os dados do arquivo. Se você não for digitar a extensão do nome do arquivo não é necessário usar o ponto que separa o nome da extensão.

Veja, a seguir, outros exemplos:

ARQVALOR	ARQBANCO	CARTA
SHARP	PRINCIPA	DIFUSE
FUNDAMEN	Y2	TEXTO

Algumas extensões, pelo seu uso constante, já são consideradas como "padrões", e quando um arquivo possui uma dessas extensões, é facilmente identificado . Veja, a seguir algumas delas:

EXTENSÃO	- SIGNIFICADO
BAS	programas escritos em BASIC
FOR	programas escritos em FORTRAN
COB	programas escritos em COBOL
ASM	programas escritos em ASSEMBLER
MAC	programas escritos em MACRO ASSEMBLER
C	programas escritos em C
REL	programa relocável
COM	programa executável em linguagem de máquina
PRN	listagem gerada na compilação de um programa
BAT	conjuntos de comandos para execução em "BATCH"
HEX	dados em hexadecimal
DAT	dados diversos
SYS	dados do sistema
ASC	dados em ASCII
TXT	textos diversos

Se você tiver um programa chamado PROG1 e ele estiver em BASIC, é conveniente chamá-lo de PROG1.BAS. Assim ficará mais fácil saber o que é o PROG1.

Algumas das extensões descritas anteriormente tem significado especial e não devem ser usadas para quaisquer arquivos. Nos capítulos seguintes você saberá o porquê disso.

NOMES DOS PERIFÉRICOS

Da mesma maneira que os arquivos, os periféricos também têm seus nomes especiais.

Periférico é tudo aquilo que está na "periferia" da Unidade Central de Processamento (UCP):o teclado, o vídeo, a impressora, o gravador cassete, o drive, etc.

Cada periférico tem associado a si um nome específico para que o computador saiba onde procurar para ler ou gravar arquivos, independentemente de a comunicação ser para enviar dados ou recebê-los da CPU.

A seguir você tem uma lista com alguns nomes de periféricos que o MSX pode aceitar ao ser ligado a um ou mais drives e/ou cartuchos.

Note que apenas quatro nomes são sempre interpretados pela CPU (CAS: , LPT: , CRT: e GRP:) ,pois estão identificados na ROM. Os demais nomes dependem de cartuchos.

NOME	PERIFÉRICO
CAT:	cartucho
CAS:	gravador cassete
LPT:	impressora
CRT:	vídeo (na SCREEN 0 ou SCREEN 1)
GRP:	vídeo (na SCREEN 2 ou SCREEN 3)
A:	drive
B:	drive
C:	drive
D:	drive — MÁXIMA CONFIGURAÇÃO (A: a F:)
E:	drive
F:	drive
NUL:	periféricos especiais
COM:	RS-232C
AUX:	periféricos especiais
CON:	console da CPU
(outros)	....

Tome cuidado ao dar nome aos seus arquivos para não misturá-los com os nomes dos periféricos. Por exemplo, chamar um arquivo de CAS.BAS pode dar muitos problemas e muita dor de cabeça no futuro!

Para indicar ao computador em qual periférico você quer que ele procure o arquivo fornecido, deve-se escrever o nome do periférico antes do nome do arquivo. Veja, a seguir, alguns exemplos:

```
CAS:ALU2
A:ARQUIVO.DAT
B:PROG1.BAS
```

Note que os dois pontos (:) servem para separar o nome do periférico do nome do arquivo.

Em resumo, sempre que quisermos que o computador armazene uma certa quantidade de informações, esta será armazenada em forma de arquivos.

Temos portanto que dar um nome a esse arquivo para que ele possa ser utilizado mais tarde.

Temos que informar, ainda, o meio físico, onde deverá ser gravado ou lido esse arquivo, dando o nome do periféco correspondente.

Somente depois de dizermos "onde" e "o que" queremos é que o computador poderá acessar os dados lá contidos.

TABELA ASCII

Para que possa haver comunicação homem-máquina, é necessário uma padronização das informações enviadas e recebidas pelo computador e pela pessoa que o utiliza.

A Tabela ASCII (American Standard Code for Information Interchange) existe para associar a cada símbolo, número ou letra que digitamos, um código numérico que o computa-

dor possa operar.

Quando digitamos "ABC" no teclado, na realidade o micro está recebendo os códigos 65, 66 e 67, que representam as letras A, B e C respectivamente (veja no apêndice I todos os códigos da Tabela ASCII).

Da mesma forma, quando o micro nos envia uma mensagem, ele o faz através de códigos que são mostrados em forma de letras para que possamos entender seu significado.

As transformações que ocorrem, desde a digitação dos dados até a sua visualização na tela do micro, passam desapercebidas por parte de quem o utiliza. Afinal, não interessa saber o que realmente acontece dentro do micro, mas apenas o resultado final.

Porém, é bom saber como os dados dos arquivos são transferidos e interpretados pelo micro, pois algumas funções do DSK-BASIC e do HB-DOS são baseadas na tabela ASCII, nos obrigando a tomar certos cuidados quando formos utilizar algumas funções.

Nessa tabela, todos os símbolos, números e letras (tanto maiúsculas como minúsculas) têm, cada uma um código diferente, começando a partir do código ASCII 32 (que representa o espaço em branco) e indo até o código ASCII 127 (que representa a função DELETE).

Os números de 0 a 31 são códigos especiais que não representam símbolos, mas sim ações que devem ser tomadas quando algum periférico os recebe. Esses códigos merecem uma atenção especial de nossa parte.

Vamos tomar, como exemplo, o código ASCII de número 7, correspondente ao BELL.

Quando enviamos este código para algum periférico (o vídeo, por exemplo) nós ouviremos um "beep". Isto porque o código ASCII 7 significa "soar o alarme" (BELL)!!

Experimente digitar o seguinte comando:

```
PRINT CHR$(7)
```

O mesmo vale para a impressora:

```
LPRINT CHR$(7)
```

Se a sua impressora tiver uma campainha, você acabará ouvindo um "beep", vindo da sua impressora. Nem todos os periféricos interpretam os códigos da tabela ASCII. O drive é um deles. Se de alguma forma (que será vista mais adiante) for enviado o código 7 (BELL) para o drive, este acabará sendo gravado no arquivo em vez de produzir um "beep". Quando você for listar este arquivo no vídeo, aí sim, ouvirá um "beep" vindo do meio da listagem e pensará que o micro está louco...

Vamos nos deter em alguns códigos apenas . A sequência de códigos &H0D e &H0A (retorno de carro e avanço de linha) significam fim de uma linha. Sempre que uma linha de programa ou um registro de um arquivo for gravado em disco na forma ASCII, a sequência &H0D &H0A será inserida em seu final para que o micro saiba que ali termina a linha.

Outro código ASCII muito importante para nós é o código &H1A (fim de arquivo). Sempre que terminamos de gravar um arquivo, o micro grava um byte a mais (&H1A) indicando que ali terminou o arquivo, e que nada mais virá adiante.

Não fique muito preocupado em entender bem o que são estes códigos. Basta você ter em mente que cada letra, símbolo, número, etc, são codificados e que certos códigos não representam um símbolo, mas sim uma ação a ser tomada. Sempre que necessário, recorra ao apêndice I.

Se você tiver esta idéia em sua mente, ficará mais fácil de entender o porquê de certos comandos que estudaremos mais adiante.

FORMATAÇÃO DE DISQUETES

Quando compramos um disquete virgem, ele vem com nada gravado. Nessa situação, ele não serve para muita coisa (nem para você se abanar nos dias muito quentes).

Ao inserir um disquete virgem no drive e tentar gravar alguma coisa, você receberá algumas mensagens de erro, dizendo que o disco está com defeito ou coisa do tipo, mas não se assuste, pois isso é normal.

Para que o drive possa acessar corretamente o disco é necessário antes "formatá-lo".

Durante a formatação, todo o disco é verificado. Cada "pedaço" do disco é testado para ver se existe ou não defeitos além de ser totalmente "limpo" e ainda para receber certos dados específicos que servirão de indicadores para o drive saber como o disco está.

Após o disquete ter sido formatado você poderá usá-lo à vontade, porque a formatação só é necessária uma única vez.

Apenas um cuidado é necessário: Tenha certeza que ao formatar um disquete não existe nada de útil gravado nele, pois após a formatação o disquete fica totalmente "limpo", sem nenhum dado gravado.

Uma boa prática é a de formatar os disquetes virgens assim que são comprados, para não ter a preocupação de saber se tal disquete foi ou não formatado quando for usá-lo.

Formatar um disquete portanto é deixá-lo apto a receber dados. Sem formatá-lo o drive não consegue utilizá-lo!!

Cada micro formata um disquete da maneira que lhe é mais conveniente, não havendo uma padronização para isso.

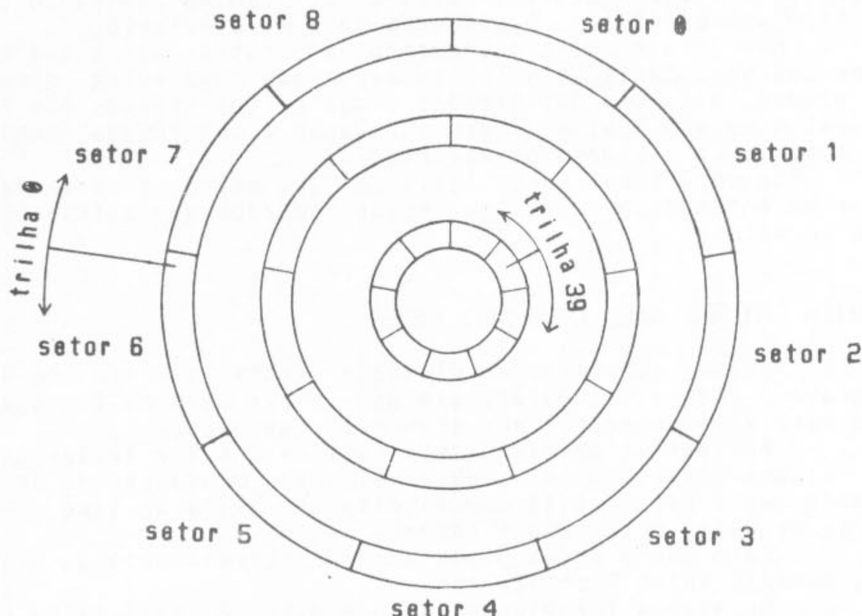
O disco é dividido em trilhas e, cada trilha é dividida em setores. A divisão do disco em trilhas e setores é o que constitui a formatação. Na figura 1.2 pode-se observar o esquema de um disquete formatado, com trilhas e setores.

A divisão do disco em trilhas depende fundamentalmente do drive utilizado. Normalmente, os drives usados por diversos micros são de um mesmo fabricante.

Sendo assim, mesmo os discos formatados em micros de marcas diferentes possuem geralmente 40 trilhas, pois esta divisão depende da precisão mecânica usada na construção do

drive.

FIGURA 1.2 - Disquete com trilha e setores.



Cada trilha do disco pode armazenar de 4 a 5 Kbytes (1 Kbyte=1024 bytes). Acessar um disco de trilha em trilha é simples, mas trabalhar com seus 4 ou 5 Kbytes não é! É devido a isso que se divide cada trilha em diversos setores.

A divisão das trilhas em setores já não depende mais tanto do drive, mas sim do programa responsável pelo acesso ao disco. Se um certo fabricante resolve subdividir uma trilha em 10 setores, então cada setor terá de 400 a 500 bytes, o que torna o trabalho com os dados muito mais fácil.

Para se dividir uma trilha em um certo número de setores é necessário gravar algumas informações entre cada um dos setores para poder distinguí-los. São essas informações inseridas na formatação que fazem, de fato, a divisão das trilhas em setores, indicando o fim de um e o início de outro. Obviamente, dos 4 ou 5 Kbytes originais de cada trilha, alguns bytes serão usados para a divisão em setores.

Na figura 1.3 pode-se observar uma representação esquemática de uma trilha dividida em 2 e 4 setores.

FIGURA 1.3 - Trilha dividida em setores.

Trilha com 2 setores:

**	setor 1	**	setor 2

Trilha com 4 setores:

**	setor 1	**	setor 2	**	setor 3	**	setor 4

Dados de controle: **

Nos micros da linha MSX (como o seu HOTBIT), tanto o DSK-BASIC quanto o HB-DOS (que serão descritos nos próximos capítulos) dividem uma trilha em nove setores de 512 bytes cada. O número de bytes disponível em cada trilha é,portanto, 9 x 512 = 4608 , ou seja, 4,5 Kbytes!

Micros de outras linhas (APPLE, TRS-80, etc) dividem uma trilha em um número diferente de setores. Isso faz com que um micro não consiga ler os dados gravados num disco de um outro micro que use uma formatação diferente.

Os dados de controle gravados entre cada setor servem, entre outras coisas, como um verificador do que foi gravado.

Assim, se por algum motivo for modificado o conteúdo de um disco e o controle não for atualizado, o micro recusará os dados lidos.

Sendo assim, utilizar discos com formatações diferentes ocasionará erros de leitura.

Com o passar do tempo, pode acontecer que a velocidade de rotação do drive se altere. Quando isso acontecer, os dados de controle podem não ser lidos no momento certo e o micro acabará acusando erro de leitura.

Para evitar esse tipo de erro, basta fazer uma revisão periódica no seu drive, recorrendo para isso, a casas especializadas.

Como você pode notar, o acesso ao disco é um tanto rígido, exigindo cuidado ao manipulá-lo para que dure muito sem precisar de novas formatações.

CAPACIDADE DE ARMAZENAMENTO

Um disquete tem uma capacidade limitada de receber dados. Essa capacidade geralmente é expressa em Kbytes (que equivale a 1024 bytes).

Se um disquete pode conter 346 Kbytes quer dizer que nele podem ser gravados 354304 caracteres.

Devido ao atual avanço tecnológico, existem vários modelos de drives, tendo os mais recentes maior capacidade, conseguindo gravar muito mais dados do que os mais antigos.

Por isso quando você tentar gravar um programa (ou um arquivo) e ele não couber no disco, você terá que trocar o disco e tentar fazer a gravação em outro.

Com o tempo você entenderá porque é conveniente possuir vários disquetes...

Você pode até fazer uma analogia do disquete com um caderno, onde a capacidade do disco é algo análogo ao número de páginas em branco do caderno.

DIRETÓRIO.

Alguns cadernos são subdivididos em várias partes, com páginas numeradas ou cores diferentes, e em seu início

existe uma espécie de índice (para as matérias, por exemplo). Ele existe para facilitar a busca de um certo bloco de páginas dentro do caderno.

Se não existisse esse índice, você teria que folhear o caderno página por página até encontrar o que deseja, perdendo muito tempo e usando o caderno mais do que o necessário, desgastando-o muito.

Em um disquete existe esse mesmo conceito, tendo um índice chamado "diretório".

No diretório existem todas as informações necessárias para que seja possível encontrarmos o que queremos, semelhante ao índice de um caderno ou de um livro, só que com muito mais informações além dos números das páginas.

Sempre que um arquivo é apagado, criado ou alterado, o diretório é atualizado de maneira a ficar sempre correto e coerente com os dados gravados no disco.

Se por qualquer azar o diretório for apagado ou avariado, não conseguiremos mais coordenar os dados do disco e, com certeza, todas as informações lá presentes não poderão mais ser acessadas na ordem correta, embaralhando tudo! Quando isso acontece (o que é muito raro) o jeito é reformatar o disquete, limpando tudo o que estava gravado!!

Para evitar tais dissabores, NUNCA insira ou retire o disquete do drive quando este estiver sendo acessado, senão você correrá o risco de interromper o acesso ao disco justamente na hora do "acerto" do diretório e aí você acabará perdendo todos os dados do disco!!

Este diretório tem o tamanho suficiente para "tomar conta" de muitos arquivos, mas pode ocorrer de você querer gravar mais arquivos (supondo que o disquete ainda tenha área disponível) e que não haja espaço no diretório para saber onde foi gravado o arquivo. Nesse caso, mesmo ainda tendo área livre, você terá que fazer a sua gravação em outro disquete.

Isso pode acontecer se você gravar muitos programas bem pequenos (que ocupam poucos bytes) e que acabam lotando o diretório sem ocupar toda a área de dados do disco. Essa é uma característica que não pode ser mudada: o espaço disponível para o diretório é fixo e imutável, sendo aconselhável fazer uma distribuição organizada dos seus programas em disquete.

CUIDADOS COM DISCOS E CóPIAS RESERVAS

Apesar do armazenamento em disco ser bastante confiável, ele ainda está longe de ser perfeito.

Você já deve ter tido desagradáveis surpresas com programas em fita cassete que são "difíceis de carregar" e outros que são praticamente " incarregáveis"!!

Com o uso do disquete essas surpresas serão bem menos frequentes, mas mas não de todo ausentes.

Por isso é muito aconselhável ter pelo menos uma cópia de cada programa gravado em disco em outro disco ou até mesmo em fita de boa qualidade.

Lembre-se que em um disquete podem ser gravados muitos programas e arquivos. Se um único disquete for danificado vários programas e arquivos serão perdidos!!

Para que um disquete fique sem condições de uso, basta dobrá-lo, amassá-lo, molhá-lo, passá-lo perto de um imã deixar cair migalhas de pão ou bolacha sobre ele, gotas de café, etc.

Por mais cuidados que se tenha ao manipulá-los, nunca estamos livres das faltas de luz, do "curioso" que gosta de por a mão em tudo, etc.

Lembre-se da LEI DE MURPHY : "Se uma coisa tem alguma chance de dar errado, certamente dará errado!"

De uma coisa você pode ter certeza: justamente aquele disquete do qual você não tem uma cópia e que é o mais importante, será o primeiro a lhe deixar na mão !! Confirmar a lei de MURPHY, na prática, é só uma questão de tempo!

Nos capítulos seguintes veremos como tirar cópias de arquivos de um disco para outro.

SISTEMAS OPERACIONAIS

Sistema Operacional é um conjunto de rotinas essenciais para que o microcomputador possa funcionar. Sem um sistema operacional, um micro não passa de um amontoado de componentes eletrônicos.

Os micros MSX já vêm de fábrica com um pequeno sistema operacional e um poderoso interpretador BASIC embutidos na ROM (a memória permanente do micro). Ambos são ativados assim que o micro é ligado.

Esse pequeno sistema operacional foi desenvolvido especialmente para os micros MSX e funciona em conjunto com o BASIC MSX. Ele é o responsável pelo vai-e-vem de informações entre os diversos periféricos: teclado, vídeo, gravador cassete, etc. O interpretador BASIC apenas decide o que fazer com os dados recebidos, sem se preocupar em saber como os dados chegaram ou como vão ser enviados.

O interpretador BASIC é algo semelhante ao gerente de uma firma: toma as decisões baseado nas informações que os boys (representados pelo sistema operacional) levam e trazem diariamente.

Todas as funções de um sistema operacional são elaboradas de acordo com o circuito eletrônico (hardware) do micro em que ele irá operar. Sendo assim, cada tipo de micro tem que ter um sistema próprio, pois seus circuitos são diferentes. Se você tentar copiar o sistema operacional de um micro que não seja padrão MSX para um HOTBIT, nada funcionará.

Os programas desenvolvidos num certo equipamento sabem de antemão qual o sistema operacional com o qual vão trabalhar, sendo então preparados para receber e encaixar os dados de acordo com os padrões pré-estabelecidos para aquele sistema operacional. Devido a isso, um programa desenvolvido em um equipamento nem sempre pode ser executado em outro.

Se quisermos fazer um programa desenvolvido para um

micro rodar em outro, temos que reestruturá-lo completamente. Isso geralmente é muito trabalhoso e nem sempre compensa.

Para diminuir a dependência entre programa e micro foram desenvolvidos sistemas operacionais com versatilidade suficiente para que, quando vistos pelo programa, pareçam sempre o mesmo, mas que possam ser instalados em micros diferentes e sem afetar o funcionamento do programa. Esses sistemas constam de duas partes: uma específica para a máquina em que ele vai funcionar e outra padronizada e independente da máquina. De um lado desse sistema operacional existe esse padrão e do outro, as rotinas que dependem do circuito eletrônico do micro no qual está rodando esse sistema operacional. Dessa maneira,para o programa, o sistema operacional é sempre o mesmo. Na realidade, para cada sistema operacional existem várias versões, desenvolvidas uma para cada equipamento. Dessa forma, um mesmo programa pode rodar em diversos micros sem nenhuma modificação.

Um desses sistemas operacionais foi batizado de CP/M (Control Program for Microcomputers) ou programa de controle para microcomputadores. O CP/M é idéia que deu certo e atualmente é o sistema operacional para o qual foi desenvolvido o maior número de programas em todo o mundo!!!. É natural, portanto, quando uma empresa lança um micro no mercado, que haja uma preocupação em produzir um sistema operacional com características do CP/M,pois assim todos os programas já desenvolvidos nesse sistema poderão ser utilizados no novo micro!

Já que o CP/M nada mais é do que um sistema operacional, seu único objetivo é gerenciar o vai-e-vem das informações entre os periféricos, cuidando da forma pela qual os arquivos serão lidos e gravados em disco, etc.

Quando se tem o CP/M num micro, a maior parte de sua memória está desocupada para poder receber o programa que irá rodar. Apenas uma pequena parcela é utilizada pelo sistema operacional para o seu próprio funcionamento.

O CP/M possui alguns comandos que permitem a você ver o que existe gravado no disco, apagar um arquivo, verificar o conteúdo de um arquivo, e assim por diante.

Um outro sistema operacional é o HB-DOS, específico para micros MSX. Ele possui muitas carcterísticas semelhantes ao CP/M mas não roda diretamente programas escritos em CP/M.

Existem muitos outros sistemas operacionais, como o MS-DOS, SOM,UNIX, etc. Cada um desenvolvido com um objetivo.

Para o HOTBIT, existem três sistemas operacionais desenvolvidos e disponíveis no mercado: o DSK-BASIC, o HB-DOS e o HB-MCP, este último, totalmente compatível com o CP/M.

Nos próximos capítulos iremos estudar detalhadamente cada um deles, suas vantagens e conveniências.

CAPÍTULO II



O QUE É O DSK-BASIC

Um dos pontos fortes dos micros MSX é a capacidade que eles tem de se expandir facilmente com o auxílio de cartuchos (e o drive é um exemplo disso). Ao se instalar um drive num micro dessa linha, um novo conjunto de instruções é liberado para uso, sem que se percam as antigas, permitindo assim maior flexibilidade no tratamento dos dados pelo programa. Além de novos comandos, alguns que já existiam e eram destinados ao gravador cassete foram expandidos e complementados de modo a acessar ao drive, tornando o HOT-BASIC muito mais versátil e poderoso.

Para poder então distinguir esses "BASIC's", com e sem drive, batizaremos de DSK-BASIC aquele BASIC com novos comandos e funções, que vamos detalhar neste capítulo.

É importante ressaltar que todos os comandos e funções do BASIC comum continuam operando com a mesma sintaxe. Por exemplo, a função VARPTR, tanto no BASIC como no DSK-BASIC é usada com a mesma sintaxe, para obter o endereço da memória em que está armazenada uma variável ou o início de um buffer de arquivo. O mesmo não ocorre com o comando RUN, que no DSK-BASIC, pode operar com duas sintaxes diferentes: a do BASIC comum e outra específica para o drive.

Para que os novos comandos funcionem perfeitamente, uma certa porção de memória é reservada exclusivamente para uso do drive, reduzindo um pouco a disponibilidade para o usuário. Ao ligar o micro, você é informado de quantos bytes pode utilizar nos seus programas. Sem o drive, você deverá ter uma tela como a da figura 2.1 .

FIGURA 2.1 - Tela inicial do HOTBIT sem drive.

HOT-BASIC versão 1.1
EPCOM [Z.F.MANAUS] 1985
Mem. Livre 28815
Ok

HOT BIT
HB-8000
EPCOM [Z.F.MANAUS] 1985

Note que há praticamente 28 Kbytes de memória livre para programas. Essa memória vai sendo ocupada conforme os programas forem sendo inseridos.

Veja no apêndice II como instalar o drive em seu HOTBIT. Ao instalar o drive e ligar o micro, sem nenhum disquete, você verá duas telas como as da figura 2.2 .

FIGURA 2.2 - Telas iniciais do HOTBIT com drive e sem disco.

```
Entre com data (D/M/A):■
```

```
HOT BIT

HB-8000

EPCOM [Z.F.MANAUS] 1985
```

Neste ponto, você deve informar a data do dia como indicado: dois dígitos para o dia, dois dígitos para o mês e dois ou quatro dias para o ano. Para separar o dia do mês e o mês do ano, você pode utilizar o sinal de menos (-) ou a barra de divisão (/). Se for informada uma data inválida, o DSK-BASIC não a aceitará, mas se você der nenhuma data, simplesmente teclando RETURN, o DSK-BASIC não se importará !!!

Após ter sido informada uma data correta ou nenhuma data, você verá a seguinte tela:

FIGURA 2.3 - Tela inicial do DSK-BASIC inicializado.

```
HOT-BASIC versão  1.1
EPCOM [Z.F.MANAUS] 1985
Mem. Livre 24455
DSK-BASIC V1.0
Ok
```

Dois pontos muito importantes precisam ser observados nas figuras anteriores:

1) Note que é mostrada uma mensagem indicando que o DSK-BASIC está instalado, permitindo assim o uso de mais comandos.

2) Observe também que diminui o número de bytes disponíveis. Isso pode causar um sério problema se você possuir programas muito grandes e que ocupam praticamente toda a memória do micro. Tais programas não podem ser carregados com o drive instalado e, ao tentar carregá-los, o micro poderá ficar "maluco", pois o programa irá sobrepor-se a uma área da memória que estava sendo ocupada pelo programa que controla o drive, fazendo-o perder o controle. Quando isso ocorrer, carregue o programa da fita cassete sem ter instalado o drive. Tente "encurtá-lo" o máximo possível e regravá-lo em fita novamente. Instale então o drive e tente ler o programa do cassete para poder salvá-lo em disco finalmente. Se mesmo assim o programa não couber, a solução será repartir o programa em duas partes distintas (se for possível) ou desacoplar o drive do micro e continuar usando apenas o gravador cassete quando precisar utilizar tal "programão"!!

Uma vez instalado o drive, ligado o micro e visualizada as telas das figuras anteriores,o DSK-BASIC estará pronto para ser usado. Uma das maneiras de se acionar o DSK-BASIC é,portanto,ligar o micro com o drive instalado mas sem nenhum disco em seu interior.

Sempre que o micro é ligado ou "resetado" e não há disco no drive, o DSK-BASIC é inicializado!

Se houver disco no drive e não existir algum sistema operacional gravado nele,então o DSK-BASIC também é inicializado. Mas, se no disco existir algum sistema operacional gravado (HB-MCP ou HB-DOS), então será dada preferência à este novo sistema e não ao DSK-BASIC. Isso será visto nos capítulos seguintes.

Dentre os vários comandos e funções adicionais, alguns são dedicados ao disco em si e não ao tratamento dos arquivos ou dos dados presentes nesses arquivos.

FORMATAÇÃO DE DISQUETES NO DSK-BASIC

Você já leu no primeiro capítulo que um disquete virgem precisa ser preparado para que possa ser utilizado. Esta preparação chama-se "formatação" e consiste basicamente numa "arrumada da casa".

Um disquete, após ter sido formatado, pode ser usado muitas e muitas vezes e se nada de catastrófico lhe acontecer nunca mais será necessário reformatá-lo.

ATENÇÃO: Não formate os discos que você recebeu junto com o drive. Eles já foram formatados e contém informações importantíssimas que não podem ser perdidas.

Para se formatar um disco virgem no DSK-BASIC, insira-o no drive e digite o seguinte comando:

CALL FORMAT (e tecle RETURN)

Após a digitação desse comando o micro "perguntará" qual o drive que contém o disco a ser formatado: A ou B, pois podem ser instalados até dois drives.

Nesse momento, aperte a letra A, informando que a formatação será feita no drive A.

Como a interface do seu HOTBIT pode aceitar tanto os disquetes de 5,25 quanto os de 3,5 polegadas, é realizada uma pergunta sobre qual dos dois drives está instalado no micro. Tecle 2 para indicar drive de 5,25 polegadas.

Após ter selecionado o drive A e o tipo do drive, é perguntado o número de faces que o drive possui : uma ou duas faces, já que existem estes dois modelos de drive. Tecle 2, indicando drive de duas faces.

Feito tudo isso, o DSK-BASIC pede para ser pressionada qualquer tecla a fim de iniciar a formatação.Tenha plena certeza de que o disco no drive A: é o que realmente deverá ser formatado, teclando então RETURN.

Após a formatação ter sido completada, o DSK-BASIC envia uma mensagem.

Agora sim, você já tem um disquete pronto para receber seus programas ou arquivos em DSK-BASIC.

Um disquete, após ter sido formatado, está completamente limpo, sem nada gravado, portanto, nunca formate um disquete que contenha alguma informação útil, a menos que queira perdê-la !!!

ESPAÇO LIVRE - DSKF(0)

Após um disquete ter sido formatado, ele está completamente limpo. A medida que o utilizamos o espaço disponível para uso vai diminuindo.

Há um comando que nos mostra o quanto ainda temos de espaço livre para gravarmos nossos programas e arquivos.

Após a formatação, o DSK-BASIC imagina o disquete como se ele fosse um "livro" e para ele o espaço que ainda resta para isso é expresso em "páginas em branco".

Numa "página" cabe uma certa quantidade de informação e enquanto esta "página" não é completada ele não utiliza outra página para o mesmo programa. Para que o DSK-BASIC possa manter um índice (diretório) indicando corretamente como estão sendo usadas as "páginas" no disquete, ele não mistura programas e arquivos diferentes em uma mesma "página".

Vamos supor que em uma "página" do disquete pudéssemos gravar 1024 bytes (1 Kbyte).

Se gravarmos um programa de apenas 100 bytes, este ocupará uma página. Mesmo que tenham sobrado 924 bytes nessa página, o DSK-BASIC não os utilizará para outro programa, pois ela já está em uso!

Se gravarmos um segundo programa de 1100 bytes, por exemplo, este ocupará uma página completa (1024 bytes) e mais 76 bytes de uma outra página, que terá seus 948 bytes restantes inutilizados.

Essa é uma característica do DSK-BASIC, e existe pa-

ra que ele possa ter um diretório (o "índice" desse "livro") simples e de fácil consulta, dinamizando muito o acesso aos dados gravados no disco. Em compensação, se gravarmos 10 programas de apenas 1 byte cada, na verdade ocuparemos 1024 bytes para cada um deles!

Podemos "ver" o funcionamento disso através da função DSKF. Ela nos mostra quantas "páginas" ainda não foram usadas e consequentemente quanto temos de espaço no disco para ser usado.

Para sabermos quantas "páginas" ainda restam no disco, digitamos:

```
PRINT DSKF(0)
```

O 0 (zero) entre parênteses indica que queremos saber o espaço livre no último drive que for acessado. DSKF(1) indica drive A:, DSKF(2) indica drive B:, e assim por diante.

Ao utilizar este comando logo após a formatação, obtém-se a capacidade total do disco, já que não há nada gravado nele ainda (figura 2.4).

FIGURA 2.4 - Espaço livre num disco recém formatado.

```
Ok
PRINT DSKF(0)
 351
Ok
```

É bom usar essa função de vez em quando para ter idéia de como o disquete está sendo utilizado, aproveitando melhor o seu espaço. Nos disquetes de 5,25" face simples cada página ocupa 512 bytes. Disquetes de 5,25" face dupla tem páginas de 1024 bytes.

SAVE

Agora que já temos um disquete em condições de uso e sabemos quanto espaco temos a disposição, vamos ver como ler e gravar programas nele.

Os comandos SAVE, LOAD, BSAVE e BLOAD sofreram pequenas modificações para servirem tanto para fitas cassetes como para os disquetes.

O comando SAVE, como você já sabe, faz com que o programa que está na memória do micro seja gravado, só que agora o seu programa será gravado no disco e não mais no cassete, como era anteriormente.

Vamos utilizar o programa a seguir como "cobaia" para vermos o "novo" comando SAVE em ação.

FIGURA 2.5 - Programa exemplo.

```
10 PRINT "MSX HOTBIT"
20 GOTO 10
```

Após tê-lo digitado, vamos gravá-lo em disco através

do comando SAVE.

Primeiramente precisamos escolher um nome apropriado: COBAIA parece ser um bom nome.

Se você quiser, poderá dar a extensão .BAS a ele, já que se trata de um programa BASIC. O nome ficaria assim:

```
COBAIA.BAS
```

Em seguida precisamos saber o nome do periférico no qual nosso programa deverá ser gravado. Como queremos gravá-lo em disco, usaremos A: como nome do periférico, mas se você possuir dois drives instalados em seu micro, poderá escolher A: para o primeiro drive ou B: para o segundo drive.

Depois de escolhido o periférico (A:), o nome completo fica assim:

```
A:COBAIA.BAS
```

Agora, é só digitar o comando completo:

```
SAVE"A:COBAIA.BAS"
```

(e teclar RETURN)

Quando voltar o "Ok", este pequeno programa que acabamos de criar já terá sido gravado em disco.

Não se espante se foram gastos apenas alguns segundos para este programa ser gravado, afinal uma das vantagens de se ter um drive é justamente a rapidez na gravação e leitura!

Essa forma de uso do comando SAVE faz com que o programa seja gravado do mesmo jeito em que ele está na memória: compactado.

Um programa compactado contém alguns códigos especiais no lugar das palavras reservadas (PRINT, GOTO, etc), ficando assim bem menor.

Em contrapartida, um programa compactado não pode ser usado pelo comando MERGE e nem pode ser lido como se fosse um arquivo sequencial (que será visto um pouco mais adiante). Para que possamos "misturar" (com o MERGE) um programa na memória com outro gravado em disco, é necessário que o programa gravado em disco esteja descompactado, ou melhor dizendo, esteja na forma ASCII.

O comando SAVE contém uma opção que,se for utilizada, faz com que o programa seja gravado em disco na forma ASCII (ou seja, sem códigos especiais e "compactado").

Vamos então gravar o nosso programa que está na memória (COBAIA.BAS) na forma ASCII utilizando o comando SAVE.

Para não "perdermos" a gravação feita anteriormente, vamos modificar um pouco o nome do nosso programa para:

```
A:COBAIA.ASC
```

Assim, ficaremos com duas versões do mesmo programa:

```
A:COBAIA.BAS
```

(na forma compactada e -->)

```
A:COBAIA.ASC        (na forma ASCII)
```

Digite, agora, o seguinte comando:

```
SAVE "A:COBAIA.ASC",A
```

Note a presença do ",A" logo após o nome do programa. É este ",A" a opção que diz que queremos gravar o programa na forma ASCII (não foi por acaso a escolha da letra "A" como opção).

Repare também que não modificamos o nome do programa propriamente dito,mas sim a extensão do nome de BAS para ASC. O nome permaneceu o mesmo (COBAIA), afinal,acabamos de salvar o mesmo programa, modificando apenas a maneira na qual ele havia sido gravado (compactado ou não compactado).

Com isso, dá para se ter uma idéia da flexibilidade em se usar as extensões dos nomes dos programas para indicar como o programa está armazenado no disco.

Agora, se quiséssemos misturar (fazer um merge) de um programa qualquer que esteja na memória do micro com o programa cobaia teríamos obrigatoriamente que usar o programa "COBAIA.ASC" e não o "COBAIA.BAS". Fica a sua escolha gravar os programas em ASCII ou não.

De maneira geral, não usamos o ",A" no comando SAVE para que o programa ocupe menos espaço, cabendo assim mais programas em um mesmo disco. O ",A" é usado apenas para aqueles programas que serão utilizados mais tarde por um comando MERGE.

A estas alturas você deve estar se perguntando como utilizar o comando SAVE com o gravador cassete já que só o utilizamos até agora com o disco. É muito simples, bastando apenas mudar o nome do periférico de A: (ou B:) que representava o disco para CAS: que representa o cassete!! Mas não se esqueça que quando usamos a fita cassete, o nome do arquivo não tem extensão e são apenas seis caracteres, no máximo, que podemos utilizar.

Para gravar o programa COBAIA em fita cassete através do comando SAVE no DSK-BASIC basta digitar o seguinte comando:

```
SAVE "CAS:COBAIA"
```

Observe que se o periférico no qual será gravado o arquivo for o cassete (CAS:), o comando SAVE não aceitará mais a opção ",A" pois este comando já grava o programa na forma ASCII. Para gravar o programa na forma compactada, existe o comando CSAVE.

Para você não se confundir com os comandos SAVE e CSAVE em como gravar seus programas, basta dar uma boa olhada na tabela a seguir, e com o tempo você praticamente "aposentará" seu gravador cassete usando-o exporadicamente devido à grande superioridade do drive sobre ele.

Além disso,muitos programas estão disponíveis em fita cassete, mas não em disco.

COMANDO	FUNÇÃO
SAVE "A:PROG",A	gravar em disco na forma ASCII o programa "PROG";
SAVE "A:PROG"	gravar em disco na forma compactada o programa "PROG";
SAVE "CAS:PROG"	gravar em cassete na forma ASCII o programa "PROG";
CSAVE "PROG"	gravar em cassete na forma compactada o programa "PROG".

Se fosse possível vermos os bytes gravados num disco veríamos a primeira linha do programa COBAIA assim:

FIGURA 2.6 - Formas de armazenamento.

PROGRAMA EM BINÁRIO

FF	14	80	0A	00	91	20	22	4D	53	58	20	48	4F	54	42	49	54	22	00
							"	M	S	X		H	O	T	B	I	T	"	

PROGRAMA EM ASCII

31	30	20	50	52	49	4E	54	20	22	4D	53	58	20	48	4F	54	42	49	54	22	0D	0A
1	0		P	R	I	N	T		"	M	S	X		H	O	T	B	I	T	"		

Se você quiser se aprofundar mais no "porquê" e no "como" dos programas serem gravados assim, sugiro a leitura do livro "Aprofundando-se no MSX" desta mesma editora, onde esse assunto foi examinado a fundo.

LOAD

Agora que já sabemos como gravar um programa em BASIC no disco , precisamos saber como recuperá-lo através do comando LOAD que também ficou um pouco mais completo.

Se você estava acostumado a utilizar o comando LOAD sem dar o nome do programa, é bom ir perdendo esse costume. Quando temos um ou mais drives instalados no HOTBIT o comando LOAD fica um pouco mais "exigente" e não dá mais aquela "colher de chá" para a nossa preguiça, pelo menos quando estamos usando o drive.

Num disquete, como você já sabe, podemos ter vários programas e arquivos gravados e precisamos informar ao comando LOAD qual deles é o que queremos carregar, sendo portanto obrigatório digitar o nome do programa.

Nem se preocupe em saber se o programa foi gravado em ASCII ou não, pois o próprio comando LOAD se encarrega de fazer isso para você e acaba lendo o programa, estando ele de uma ou outra forma.

Se você quiser agora carregar o programa COBAIA.BAS

do disquete digite o comando:

```
LOAD "A:COBAIA.BAS"
```

Atente para o fato de que, independentemente do programa ter sido gravado ou não através da opção ",A" do comando SAVE, o programa na memória ficará sempre na forma compactada após ter sido carregado por qualquer comando, LOAD ou CLOAD !!

Se você quiser carregar o programa da fita cassete deverá usar CAS: no lugar do A: , da mesma maneira que no comando SAVE.

```
LOAD "CAS:COBAIA"
```

Mas não se esqueça: use o comando LOAD para carregar programas gravados pelo comando SAVE e CLOAD para aqueles gravados pelo CSAVE (no caso do cassete).

Agora já temos conhecimento suficiente para passarmos os programas BASIC da fita cassete para o disco:

1) Carregue o programa da fita cassete com o comando CLOAD ou LOAD"CAS:",dependendo de como ele tenha sido gravado originalmente.

2) Após ter sido carregado, dê uma verificada para ver se está tudo OK, fazendo as alterações que forem necessárias.

3) Grave o programa em disco com o comando SAVE"A:NOMEPROG.EXT" substituindo NOMEPROG.EXT pelo nome que você achar mais conveniente.

4) Se este programa for usado mais tarde em um comando MERGE, utilize a opção ",A" no comando SAVE

MERGE

O comando MERGE serve para misturar o programa que já está na memória do HOTBIT com um outro programa que pode estar tanto no disco como também em fita cassete.

Este comando não teve uma alteração muito profunda em sua sintaxe, e como outros comandos, o MERGE apenas passou a aceitar o disco.

O programa que for lido , da fita ou do disco,deve ter sido obrigatoriamente gravado no formato ASCII.

As linhas de programas que são inseridas pelo comando MERGE tem prioridade em relação às linhas que já estão na memória do micro.É como se você estivesse digitando as linhas lidas pelo MERGE.

É muito comum, com a ajuda deste comando, termos várias sub-rotinas muito utilizadas num disco e todas elas gravadas em ASCII, assim quando fizermos algum programa que precise daquela sub-rotina, basta dar um MERGE e pronto !!!

BLOAD & BSAVE

De maneira semelhante, os comandos BLOAD e BSAVE também passaram a "aceitar" o novo periférico.

Porém, não houve uma mudança significativa nem na forma e nem na função exercida pelos comandos, mas apenas o acréscimo de uma opção, proporcionando assim um recurso a mais ao usuário.

Se você quiser passar os seus programas em ASSEMBLY da fita cassete para o disco, basta carregá-los com o comando:

```
BLOAD "CAS:"
```

E gravá-los com o comando:

```
BSAVE "A:PROGRAMA.OBJ",início,fim,execução
```

Os endereços de início, fim e de execução do programa não devem ser esquecidos! Grave o programa com o nome que você achar melhor.

Preste atenção se o programa que você vai carregar da fita não vai se sobrepor à área da memória ocupada pelo drive. Se isto ocorrer, não há como carregar o programa da fita com o drive instalado e, consequentemente, você terá que continuar a usar esse programa em fita cassete.

Com programas escritos em BASIC é relativamente fácil alterarmos o seu tamanho, juntado linhas, economizando alguns comandos, etc mas com programas em ASSEMBLY esta tarefa é extremamente difícil senão impossível. Portanto, se algum programa em ASSEMBLY fizer o seu HOTBIT ficar "maluco" provavelmente ele invadiu uma região da memória que não devia e só lhe resta utilizar este programa sem o drive.

O que de melhor existe nestes dois comandos é que agora, além de ler e gravar os dados contidos numa região da memória, eles podem acessar a memória de vídeo (VRAM) também. Como resultado, podemos ler e gravar uma imagem do vídeo.

Para diferenciar se queremos ler ou gravar dados da memória principal ou da memória de vídeo usamos a opção ",S" (de SCREEN).

Vamos supor que você tenha um programa em BASIC que desenhe uma paisagem no vídeo. Após a paisagem ter sido desenhada, pode-se colocar um comando no programa de forma a gravar tal imagem. Quando você quiser refazer a paisagem, basta recarregá-la do disco em vez de rodar o programa que a gerou.

Para gravar uma imagem, use o seguinte comando:

```
BSAVE "A:TELA",início,fim,S
```

Onde:

TELA - é o nome do arquivo.
início - é o endereço inicial da memória de vídeo que deverá ser gravada. Este endereço varia de acordo com a te-

la (SCREEN) em uso

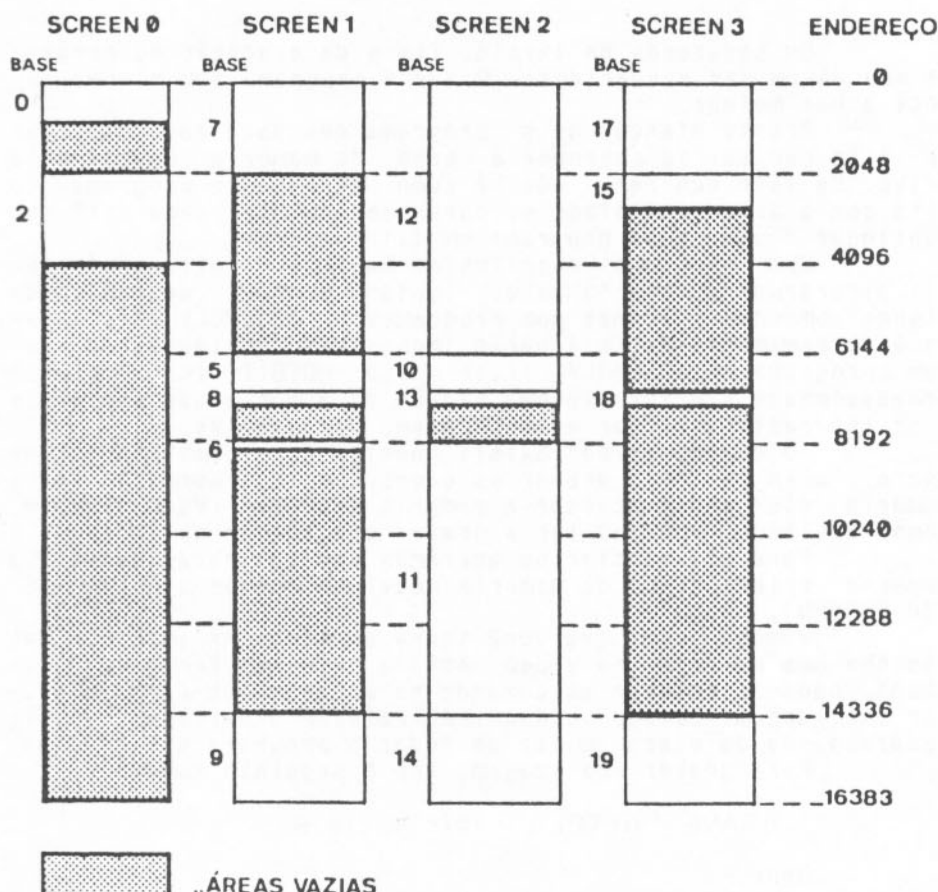
fim - é o endereço final da memória de vídeo que deverá ser gravada, dependendo também do último comando SCREEN que foi executado

S - é a opção que permite ser usada a memória de vídeo e não a memória principal.

Para ler uma imagem, use a forma:

```
BLOAD "A:TELA",S
```

Veja a seguir como está organizada a memória de vídeo, de acordo com o modo SCREEN selecionado. A estrutura da VRAM e o funcionamento do VDP são comentados detalhadamente no livro "APROFUNDANDO-SE NO MSX" desta mesma editora.



FILES & LFILES

Até agora já sabemos como passar os programas escritos em BASIC e também aqueles que foram escritos em linguagem de máquina, da fita cassete para o disco, e vice-versa.

Depois de um certo tempo de uso do drive você acabará tendo vários disquetes e em cada disquete vários programas e arquivos. É natural acabar esquecendo o que foi gravado em em um ou outro disquete.

O comando FILES existe justamente para nos ajudar nessas horas. Ele nos mostra o nome de todos os programas e arquivos que estão gravados no disco, bastando para isso digitar o seguinte comando:

```
FILES                (e teclar RETURN)
```

Se você desejar uma cópia impressa, utilize o comando LFILES. Ele é análogo ao FILES, só que a listagem sairá na impressora e não no vídeo.

Veja na figura a seguir um exemplo do comando FILES em um disco com vários arquivos gravados.

FIGURA 2.7 - Exemplo de FILES.

```
Ok
FILES
JOGO1   .BAS JOGO2   .BAS TEXTO   .TXT
SKYJAGUA.GAM PINGUIM .GAM LIXO
COBAIA  .BAS COBAIA  .ASC
Ok
```

Se você utilizar o comando FILES e não houver nenhum arquivo em disco, será enviada a seguinte mensagem :

FIGURA 2.8 - Mensagem de arquivo inexistente.

```
Ok
FILES
Arquivo não existe
Ok
```

Outra característica do comando FILES é que, de acordo com o último comando SCREEN e WIDTH utilizado, ele mostra uma, duas, três e até mais colunas com os nomes dos arquivos, dependendo de quantos cabem numa mesma linha do vídeo. Experimente dar vários comandos WIDTH com argumentos diferentes e utilizar o comando FILES.

O comando FILES pode ser usado para saber se um programa em específico existe ou não no disco. Para isso, digite o nome do programa logo após o comando FILES. Se o arquivo existir, seu nome será mostrado no vídeo. Caso contrário, a mensagem de arquivo inexistente será apresentada.

Observe a figura 2.9 .

FIGURA 2.9 - Uso do FILES para achar um programa.

```
Ok
FILES "JOGO1.BAS"
JOGO1   .BAS
OK
FILES "JOGO5.BAS"
Arquivo não existe
Ok
```

Essa maneira de se usar o comando FILES é muito útil quando sabemos o nome do programa, mas não sabemos onde ele está.

Existem casos em que queremos saber quais os programas em BASIC que estão gravados no disco. Se for usado o comando FILES, simplesmente aparecerá no vídeo tudo o que lá está, ficando um pouco incômoda a procura destes programas entre os vários que estão sendo mostrados. Existem dois caracteres especiais para facilitar esse trabalho de procura: o ponto de interrogação (?) e o asterisco (*).

O ponto de interrogação, no meio do nome do programa, significa que naquela posição não se sabe qual letra deveria estar, servindo, portanto, qualquer uma.

Podemos então digitar FILES "?OBAIA.BAS" que serão mostrados todos os arquivos que tenham OBAIA.BAS em seu nome , seja qual for a primeira letra do nome (XOBAIA.BAS , TOBAIA.BAS , etc).

Se digitarmos FILES "COBAIA.???", serão mostradas todas as "versões" do programa COBAIA.

FIGURA 2.10 - Uso do FILES com ? .

```
Ok
FILES "COBAIA.???"
COBAIA  .BAS COBAIA  .ASC
Ok
```

O asterisco (*) é outro caracter especial e significa que vale qualquer sequência de letras no nome (não apenas uma letra, como no caso do ponto de interrogação).

FIGURA 2.11 - Uso do FILES com * .

```
OK
FILES "*.BAS"
JOGO1   .BAS JOGO2   .BAS COBAIA  .BAS
OK
```

Quando digitamos apenas o comando FILES , equivale a digitar FILES "*.*" ou FILES "????????.???", sendo então mostrados todos os arquivos presentes no disco.

COPY

Você já leu no capítulo 1 e na prática irá confirmar a necessidade de ter uma cópia de seus programas e arquivos. O DSK-BASIC tem um comando próprio para isso: o COPY .

Com ele você pode copiar um arquivo para o mesmo disco com outro nome, ou copiar um arquivo de um disquete para outro.

Mesmo que você tenha apenas um drive instalado no seu micro, o DSK-BASIC poderá "pensar" que tem dois drives, sendo um deles o A: e o outro o B:.

Para copiar o velho COBAIA.BAS de um disco para o outro há duas maneiras: a primeira seria a de carregar o programa na memória do micro atavés do comando LOAD, trocar o disco e gravar o programa com o comando SAVE ; a segunda maneira seria digitar o seguinte comando:

```
COPY "A:COBAIA.BAS" TO "B:COBAIA.BAS"
```

Desse jeito, embora só exista um drive instalado, o DSK BASIC fará a cópia de um disco para o outro, simulando assim o mesmo drive como A: e B: .

Ao receber este comando, o DSK-BASIC carrega uma parte do programa em uma área de memória própria para este fim, sem prejudicar o programa que está na memória. Pede para você trocar o disco (retirar o disco original e pôr o disco que irá receber a cópia) e digitar qualquer tecla. Em seguida é gravada no novo disco (como se fosse o drive B:) aquela parte que foi lida no programa. Após a gravação, se o programa for muito grande e não der para ser todo copiado, então o DSK-BASIC pede para ser colocado o disco original (A:) novamente e espera você digitar qualquer tecla. Um novo trecho é então lido. Após a leitura, novamente será pedido o disco que irá receber a cópia (B:) e você terá que digitar qualquer tecla.

Assim vai indo, copiando o arquivo pedaço por pedaço, sem alterar o programa que porventura esteja na memória do micro, mas o troca-troca de disquetes (simulando o drive A: e o drive B:) é muito grande, consumindo assim muito tempo e trabalho.

Portanto, se você quiser tirar cópia dos programas em BASIC, é preferível carregá-lo na memória do micro através do comando LOAD, trocar o disquete e gravar o programa no novo disquete através do comando SAVE . Será muito mais rápido e menos trabalhoso.

Neste comando, também é possível o uso do ? e do *. Por exemplo:

```
COPY "A:*.BAS" TO "B:"
```

KILL

O comando KILL existe para podermos "matar" aquele programa que não nos serve mais. Este comando simplesmente apaga o programa do disquete, liberando o espaço ocupado por ele para os outros programas que venham a ser gravados.

Assim, quando você for gravar um programa e receber uma mensagem de erro dizendo que não há mais espaço livre no disco, verifique com o comando FILES se por acaso não existe algum programa antigo que não lhe serve mais para nada. Se e-xistir, apague-o com o comando KILL e, quem sabe, haja espaço suficiente para receber o programa que você queria gravar.

Para apagar o programa A:COBAIA.ASC basta dar o se-guinte comando:

```
KILL "A:COBAIA.ASC"
```

Você pode usar o ponto de interrogação (?) ou o as-terisco (*) da mesma maneira que no comando FILES. Por exem-plo, para apagar todos os programas .BAS, digite o seguinte comando:

```
KILL "A:*.BAS"
```

Para "limpar" todo disquete, digite:

```
KILL "A:*.*"
```

Cuidado com o uso do comando KILL, pois um arquivo que tenha sido apagado do disco nunca mais poderá ser recu-perado.

NAME

Podemos mudar o nome de um arquivo do disco através do comando NAME.

Geralmente este comando é usado em programas que al-teram dados de um arquivo, mudando-o de nome após sua execu-ção para identificar se aquele arquivo já foi "mexido" por a-quele programa; ou então quando você quiser mudar o nome de um certo programa para um outro nome mais sugestivo. Exempli-ficando, temos:

```
NAME "A:LIXO2" AS "A:COBAIA.BAS"
```

Neste comando, também é permitido o uso do ? e *:

```
NAME "A:*.BAS" AS "A:*.ASC"
```

Em qualquer caso, no comando NAME, é usado o novo no-me antes do "AS" e o velho depois do "AS".

AUTOEXEC.BAS

O DSK-BASIC possui um nome de programa que é especial:AUTOEXEC.BAS .

Sempre que o DSK-BASIC é inicializado, da maneira descrita no começo desse capítulo ele procura no disco a existência deste programa. Se ele não existir, então é passado para você o controle do micro, sendo visualizado o tradicional "Ok", mas se for encontrado algum programa com o nome AUTOEXEC.BAS , então ele será carregado na memória do micro e executado, não sendo mostrado "Ok".

Com esta facilidade, você pode modificar o nome daquele programa que você sempre utiliza (através do comando NAME) para AUTOEXEC.BAS, assim sempre que você ligar o micro com aquele disquete no drive, tal programa já será carregado sem o trabalho de se digitar algum comando.

Uma outra utilidade, seria a de se fazer um programa do tipo "MENU" que mostra no vídeo os programas que estão naquele disquete. Este programa espera que seja teclada a opção desejada, executando então o programa pedido.

Isso é muito prático num disquete de jogos. Por exemplo, você teria no disquete os dez jogos que você mais gosta e teria o programa AUTOEXEC.BAS ,que é o programa menu.

Ao ser ligado o micro com este disquete,aparecerá um menu com os dez jogos gravados, esperando você teclar a opção desejada, sem que você tenha que utilizar FILES para ver os nomes dos jogos, e BLOAD para carregar e executar um certo jogo, deixando esta tarefa para o programa menu.

O DSK-BASIC só reconhece a presença deste arquivo quando é inicializado com um disco dentro do drive. Quando você vê o "Ok" do DSK-BASIC,não adianta mais colocar um disco com o arquivo AUTOEXEC.BAS, porque o DSK-BASIC não o interpretará. Apenas quando é dado RESET ou quando o micro é ligado é que o DSK-BASIC verifica no disco a presença de tal arquivo.

TIPOS DE ARQUIVOS NO DSK-BASIC

Além dos vários comandos já vistos para tratamento dos programas e arquivos como um todo, existem outros comandos e várias funções para o tratamento dos dados que estão contidos nos arquivos (campos).

No DSK-BASIC existem dois tipos de arquivos: Arquivos Sequenciais e Arquivos Randômicos (ou de acesso direto).

Nos arquivos sequenciais, tanto a leitura quanto a gravação só podem ser feitas sequencialmente, uma após a outra, que já não acontece com os arquivos de acesso direto,onde um registro pode ser lido ou gravado em qualquer ordem.

No arquivo sequencial, os registros são gravados sequencialmente e podem ser de tamanhos diferentes. Na leitura desses arquivos, cada registro só pode ser acessado após o seu anterior já ter sido lido, e uma vez lido não é possível voltar atrás para acessá-lo novamente.

Nestes arquivos, os registros estão separados por um delimitador, indicando que ali terminou o registro e que em seguida deverá vir o próximo. Este delimitador é a sequência &H0D &H0A que são os códigos ASCII do retorno do carro e o avanço de linha.

Quando um registro é gravado, o DSK-BASIC grava automaticamente estes dois bytes para separar o registro que acabou de ser gravado do próximo. Durante a leitura de um registro, o DSK-BASIC lerá tantos caracteres quantos necessários até encontrar a sequência &H0D &H0A e todos os caracteres lidos (exceto o &H0D &H0A) constituirão então o registro.

Quando gravamos um programa com a opção ",A" do comando SAVE, este é gravado como se fosse um arquivo sequencial, sendo cada linha deste programa um registro. Os registros estão separados pela sequência &H0D &H0A.

Para criarmos um arquivo sequencial, precisamos primeiro abri-lo através do comando OPEN.

Figura 2.12 - Comando OPEN.

```
Ok
OPEN "A:TESTE.SEQ" FOR OUTPUT AS #1
Ok
```

Através do comando da figura 2.12 foi criado um arquivo de nome TESTE.SEQ no drive A. A sequência FOR OUTPUT indica que o arquivo será aberto como saída e será um arquivo sequencial, portanto serão gravados dados neste arquivo para uso posterior.

A sequência AS #1 quer dizer que de agora em diante esse arquivo será referenciado pela sequência #1 e não pelo nome TESTE.SEQ no decorrer do programa. O DSK-BASIC ao receber este comando, verifica se já existe este arquivo no drive A. Se já existir,ele será então apagado. A partir daí, não existe mais o arquivo TESTE.SEQ podendo ser criado um novo arquivo com nenhum dado gravado. Devido a essa característica, é preciso um pouco de atenção no uso do comando OPEN com FOR OUTPUT.

Uma vez aberto o arquivo, utilizamos o comando PRINT #1 para gravar dados, da mesma maneira que utilizaríamos o PRINT para mandar dados para o vídeo.

FIGURA 2.13 - O comando PRINT # .

```
Ok
PRINT#1, "MSX"
Ok
PRINT#1, "FIM"
Ok
```

Uma vez gravados todos os dados necessários, é preciso fechar o arquivo para que o DSK-BASIC possa dar uma acertada no diretório e liberar o arquivo TESTE.SEQ para uso.

Na verdade, quando mandamos gravar um dado num arquivo, esse dado não vai diretamente para o disco. Ele vai para uma região da memória chamada buffer e fica lá até que haja uma quantidade suficiente de dados para serem gravados.

O DSK-BASIC faz isso porque leva muito tempo para se gravar um byte no disco e se, para cada informação enviada através do comando PRINT # ,o DSK-BASIC fosse acessar o disco, o programa ficaria lento e o drive seria excessivamente utilizado. Sendo assim, os dados vão sendo guardados no buffer até que haja uma quantidade suficiente para que valha a pena acessar o disco.

O comando CLOSE diz ao DSK-BASIC que o arquivo não será mais acessado, e nesse caso os dados que ainda estão no buffer deverão ser gravados.

Se você retirar o disquete do drive ou desligar o micro com algum arquivo aberto sem ter dado o comando CLOSE , correrá o risco de perder algumas informações que ainda estavam no buffer,ficando assim incompleto seu arquivo. Sem falar no fato de que o diretório do disco também só é atualizado quando utilizamos o comando CLOSE.

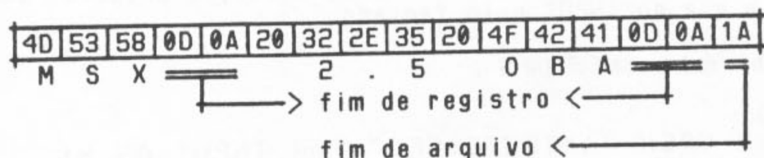
O comando CLOSE faz com que o DSK-BASIC faça os acertos necessários tanto no arquivo quanto no diretório. Veja um exemplo completo:

FIGURA 2.14 - O comando CLOSE .

```
10 OPEN "A:TEXTE.SEQ" FOR OUTPUT AS#1
20 PRINT #1,"MSX"
30 PRINT #1,5/2;"OBA"
40 CLOSE #1
Ok
```

Na figura 2.15 você tem uma representação dos dados gravados no arquivo sequencial pelo programa anterior.

FIGURA 2.15 - Dados gravados num arquivo.



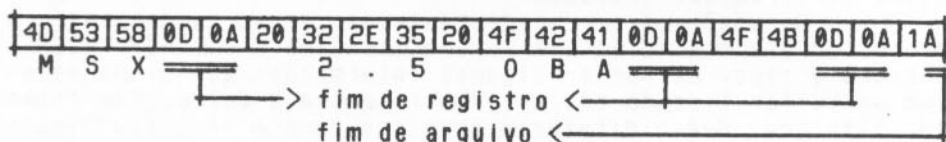
Se após gravar um arquivo você desejar acrescentar a ele alguns registros, não poderá usar FOR OUTPUT, senão o arquivo será apagado para dar lugar a um novo. Em lugar do FOR OUTPUT deve-se usar FOR APPEND.

FIGURA 2.16 - Uso do FOR APPEND.

```
Ok
10 OPEN "A:TESTE.SEQ" FOR APPEND AS #1
20 PRINT #1,"OK"
30 CLOSE #1
Ok
```

No exemplo anterior, foi adicionado mais um registro no final do arquivo.

FIGURA 2.17 - Registro acrescido ao arquivo com FOR APPEND.



Se for utilizado FOR APPEND e o arquivo não estiver no disco, então é mostrada uma mensagem de erro.

```
OPEN "ARQ1.BAC" FOR APPEND AS #1
Arquivo não existe
Ok
```

Para a leitura de um arquivo sequencial, usa-se FOR INPUT no comando OPEN.

```
OPEN "A:TESTE.SEQ" FOR INPUT AS #1
```

Neste caso, se o arquivo TESTE.SEQ não se encontrar no drive A, o DSK-BASIC interromperá a execução do programa e emitirá uma mensagem de erro, como quando é usado FOR APPEND.

Uma vez aberto o arquivo, usa-se o comando INPUT #1 para se efetuar a leitura dos dados presentes no arquivo da mesma maneira que no INPUT pelo teclado.

FIGURA 2.18 - Lendo dados com FOR INPUT .

```
Ok
10 OPEN "A:TESTE.SEQ" FOR INPUT AS #1
20 INPUT #1,X$
30 PRINT X$
40 CLOSE #1
Ok
RUN
MSX
Ok
```

No exemplo anterior, a variável X$ recebeu o texto "MSX" que estava gravado no primeiro registro do arquivo de nome TESTE.SEQ .

O comando INPUT # deve receber os dados na mesma ordem e do mesmo tipo que foram gravados pelo comando PRINT # .

Se num registro foram gravados campos numéricos junto com campos alfanuméricos, então o comando INPUT # deve saber a sequência exata na qual se encontram para não acabar recebendo nomes de pessoas quando esperava receber um certo número, por exemplo.

Quando for gravar campos alfanuméricos em um arquivo sequencial, deve-se dar atenção em como separá-los, senão ao se ler os dados através do comando INPUT #, , poderá ocorrer de se ler mais ou menos campos do que foram originalmente gravados !!!

Vamos imaginar duas variáveis:

```
A$= "Rua Barão Vermelho, casa 1"      e
B$= "São Paulo"
```

O comando Print #1,A$;B$ gravará a seguinte imagem no disco:

```
Rua Barão Vermelho, casa 1 São Paulo
```

Se for usado o comando INPUT #1,C$,D$ fará:

```
C$="Rua Barão Vermelho"      e
B$= "casa 1 São Paulo"
```

Isso não é o que se deseja e acr tece pelo fato de que a vírgula (,) é interpretada pelo DSK BASIC como um separador de campos. Para contornar isso, podemos gravar as aspas (") diretamente, fazendo com que cada campo do registro fique entre duas aspas. Por exemplo:

```
PRINT #1,CHR$(34);A$;CHR$(34);CHR$(34);B$;CHR$(34)
```

Com este artifício, a variável A$ ficou entre duas aspas (CHR$(34)) e a variável B$ também.

Quando for usado, o comando INPUT #1,C$,D$ fará com que C$ receba "Rua Barão Vermelho, casa 1" e D$ receba "São Paulo", que é o correto.

Quando os dados a serem gravados forem dados numéricos deve-se utilisar o ponto e vírgula para separá-los, pois o DSK-BASIC saberá como achá-los posteriormente quando for utilizado o comando INPUT # .

Se utilizarmos este comando e não existir mais nenhum dado a ser lido no arquivo, o DSK-BASIC envia uma mensagem de erro. Por exemplo:

```
INPUT #1,A
Fim do arquivo
Ok
```

Como ninguém é mágico ou advinho, não se pode prever se existe ou não mais dados a serem lidos no arquivo. Por isso existe uma função que nos informa se ainda resta alguma informação gravada no arquivo. Essa função chama-se EOF.

Quando abrimos um arquivo e usamos o comando INPUT # o DSK-BASIC verifica se ainda resta algo a ser lido e posiciona o EOF. Se existirem dados no arquivo, EOF devolverá o valor 0 (falso), caso contrário devolverá -1 (verdadeiro).

Para utilizar este comando é necessário informar o número com o qual o arquivo foi aberto logo após a função EOF e entre parênteses.

Sempre que for usado o comando INPUT #, deve-se utilizar a função EOF antes, para garantir que realmente existe algum dado a ser lido. Note também que esta função deve ser usada apenas com arquivos sequenciais.

FIGURA 2.19 - Uso do EOF.

```
Ok
IF EOF(1) THEN PRINT "ACABOU"
Ok
INPUT #1,A
IF EOF(1) THEN PRINT "ACABOU"
ACABOU
Ok
CLOSE
Ok
```

ARQUIVOS RANDÔMICOS

Com arquivos randômicos, a idéia de registro muda um pouco. Não existem delimitadores entre os registros, obrigando-os a ter um tamanho fixo, definido na abertura do arquivo. Os campos são também de tamanhos fixos e pré-definidos assim que o arquivo é aberto. Desse jeito, com os registros sempre do mesmo tamanho, torna-se fácil para o DSK-BASIC localizar qualquer registro dentro de um arquivo. Por conseguinte, pode-se ler os registros em qualquer ordem, não sendo obrigatório o acesso sequencial.

Para se abrir um arquivo randômico, omite-se o tipo no comamdo OPEN. Por exemplo:

```
Ok
OPEN "TESTE" AS #1 LEN=24
Ok
```

No exemplo anterior, não foi especificado nem FOR OUTPUT, nem FOR INPUT e nem FOR APPEND , assumindo assim arquivo randômico.

Se o arquivo já existir no disco, então ele será aberto sem perder os dados lá gravados. Se não existir, então será criado um novo arquivo.

A sequência LEN=24 indica que cada registro contém 24 bytes. Se ele for omitido será assumido 256 bytes.

Após a abertura do arquivo, ele poderá ser utilizado tanto para leitura quanto para gravação. Antes porém, é preciso especificar como estarão os campos dentro do registro, e isto é feito através do comando FIELD.

O comando FIELD diz ao DSK-BASIC quantos bytes terão cada um dos campos. Por exemplo:

```
Ok
FIELD #1,1 AS A$, 3 AS B$, 20 AS C$
Ok
```

Com esse comando, foi reservado 1 byte para a variável A$, 3 para a variável B$ e 20 para C$. Neste caso, o registro contém 24 bytes (1+3+20) e este foi especificado no comando OPEN anterior.

Quando for efetuada uma leitura deste arquivo, o DSK-BASIC lerá sempre de 24 em 24 bytes, independentemente do que lá está gravado, deixando o primeiro byte em A$, os próximos 3 bytes em B$ e os 20 bytes restantes em C$.

Quando é utilizado o comando FIELD, as variáveis que aí aparecem, são colocadas em um lugar próprio para os arquivos, na memória do computador. Sendo assim, estas mesmas variáveis não podem aparecer em comandos como LET, INPUT, etc, senão elas deixarão de estar na área reservada para os arquivos e passarão para a área de memória reservada para as variáveis. Então não se conseguirá mais acessar os dados do disco pois não haverá mais variáveis para receber tais dados.

Como não se pode mais usar o LET e o INPUT nas variáveis que estão no comando FIELD, novos comandos foram acrescentados para permitir a essas variáveis receber dados.

Os comandos RSET e LSET alinham à direita e à esquerda respectivamente um dado alfa-numérico em uma variável. Por exemplo:

```
Ok
LSET C$="Avenida Paulista"
Ok
```

Com isso, a variável C$ ficaria com o seguinte conteúdo:

FIGURA 2.20 - Variável C$ na memória.

41	76	65	6E	69	64	61	20	50	61	75	6C	69	73	74	61	20	20	20	20
A	v	e	n	i	d	a		P	a	u	l	i	s	t	a				

Um outro exemplo é:

```
RSET C$="Avenida Paulista"
```

Com ele, a variável C$ ficaria assim:

20	20	20	20	41	76	65	6E	69	64	61	20	50	61	75	6C	69	73	74	61
				A	v	e	n	i	d	a		P	a	u	l	i	s	t	a

Note que nos dois casos, a variável C$ contém sempre 20 bytes de tamanho, pois ela assim foi definida no comando FIELD.

Os comandos RSET e LSET alinham os dados na variável completando com brancos os bytes não ocupados e truncando os bytes excedentes.

Você deve ter reparado que no comando FIELD apenas variáveis alfa-numéricas podem ser especificadas. Para gravarmos números em um arquivo randômico, existem 3 funções que "transformam" os números em dados alfa-numéricos de tamanho fixo: MKI$, MKS$ e MKD$.

MKI$: é usado para variáveis inteiras.
MKS$: é usado para variáveis de precisão simples.
MKD$: é usado para variáveis de precisão dupla.

Estas 3 funções convertem um número em uma variável alfa-numérica de 2, 4 ou 8 bytes, independentemente do valor do número. Após esta transformação,eles podem ser movidos para o buffer do disco através de LSET ou RSET.

A seguir temos um exemplo de como mover dados numéricos e alfa-numéricos para variáveis de arquivo randômico.

FIGURA 2.21 - Utilização do MKI$, MKS$ e MKD$.

```
Ok
10 OPEN "A:NOMES" AS #1 LEN=22
20 FIELD #1, 20 AS NM$, 2 AS ID$
30 INPUT "Qual o seu nome ";A$
40 INPUT "Qual a sua idade";I%
50 LSET NM$=A$
60 LSET ID$=MKI$(I%)
Ok
```

* Na linha 10 o arquivo foi aberto, sendo indicado que cada registro contém 22 bytes.
* Na linha 20 foi definido o registro como tendo 20 bytes para a variável NM$ e 2 bytes para ID$.
* Nas linhas 30 e 40 foram recebidos os dados que deverão ser gravados. Note que NÃO são as mesmas variáveis que estão no comando FIELD.
* Na linha 50 foi transferido o conteúdo da variável A$ para a variável do disco NM$, alinhado à esquerda e na linha 60 foi transferido o valor da variável I% para ID$ no disco.

Após estas transferências, os dados ainda não foram gravados no disco, eles foram apenas posicionados. Para realmente gravá-los é necessário usar o comando PUT. Por exemplo:

```
PUT #1,1
Ok
```

O número do registro poderá ir de 1 até mais de 4000000 e no exemplo anterior foi utilizado 1. Porém, é desaconselhável dar números aleatórios ou mesmo espaçados entre si. Quando você for elaborar um programa que grava um arquivo randômico, é bom imaginar um meio de fazer com que as gravações sejam feitas em ordem crescente do número do registro e sem que haja falhas na sequência. Por exemplo, gravando os registros 1, 2, 3, 4, e assim por diante em vez de gravá-los desordenadamente: 1, 10, 8, 2, 4, etc.

Pode-se tanto ler quanto gravar registros em um arquivo randômico sem ter que fechar e abrir novamente o arquivo. A leitura de um determinado registro é feita através do comando GET. Por exemplo:

```
GET #1,2
```

O registro lido no exemplo anterior é o de número 2 e supõe-se que tenha sido previamente gravado através do comando PUT.

Após a execução do comando GET, os dados estão disponíveis para uso naquelas variáveis que foram definidas no comando FIELD.

Pode-se usar PRINT ou passá-las para outras variáveis. Apenas aqueles números que foram transformados em variáveis alfa-numéricas é que precisam ser convertidos em números novamente. Para isso existem 3 funções:CVI, CVS e CVD.

Elas convertem os números que foram transformados pelo MKI$, MKS$ e MKD$ respectivamente. Por exemplo.

FIGURA 2.21 - Uso do CVI .

```
Ok
10 GET# 1,1
20 PRINT NM$,CVI(ID$)
Ok
```

Na linha 10 foi efetuada a leitura do primeiro registro do arquivo e na linha 20 foi mostrado o nome da pessoa e a sua respectiva idade, mostrando a variável NM$ diretamente e convertendo o número que estava em ID$ através da função CVI.

Para se ter uma idéia deste conjunto todo de comandos e instruções, analise o programa a seguir. Ele não é um programa muito prático mas se for bem entendido lhe dará muitas dicas de como usar arquivos randômicos.

```
10 OPEN "A:PRODUTOS.TXT" AS #1 LEN=31
20 FIELD #1,3 AS CD$,20 AS NM$, 8 AS PR$
30 FIELD #1,2 AS UT$,29 AS LX$
40 IF LOF(1)=0 THEN UL%=1: GOTO 70
50 GET #1,1
60 UL%=CVI(UT$)
70 CLS : PRINT "Cadastro de produtos"
80 PRINT "--------------------" : PRINT
```

```
90 PRINT "1....Incluir produtos"
100 PRINT "2....Consultar produtos"
110 PRINT "3....FIM" : PRINT
120 INPUT"Opção";OP
130 ON OP GOSUB 1000,2000,3000
140 GOTO 70
1000 REM INCLUSAO
1010 CLS : PRINT"Inclusão" : PRINT
1020 INPUT "Código do produto";CP$
1030 IF LEN(CP$)<>3 THEN 1020
1040 INPUT "Nome do produto";NP$
1050 INPUT "Preço do produto";PP#
1060 LSET CD$=CP$
1070 LSET NM$=NP$
1080 LSET PR$=MKD$(PP#)
1090 PUT #1,UL%+1
1110 UL%=UL%+1
1120 RETURN
2000 REM CONSULTA
2010 CLS
2020 INPUT "Consultar qual código de produto";CP$
2030 FOR I%=2 TO UL%
2040 GET #1,I% : IF CD$=CP$ THEN 2080
2050 NEXT I%
2060 PRINT "Produto ";CP$;" não encontrado"
2070 GOTO 2120
2080 PRINT "Produto ";CD$ : PRINT "Nome    ";NM$
2090 PRINT "Preço ";USING"###,###,###.##";CVD(PR$)
2100 PRINT "Tecle qualquer tecla."
2110 X$=INPUT$(1)
2120 RETURN
3000 REM FIM
3010 LSET LX$=STRING$(29,32)
3020 LSET UT$=MKI$(UL%)
3040 PUT #1,1
3050 CLOSE
3060 END
```

Os pontos mais importantes são:

* É possível usar mais de um comando FIELD para o mesmo arquivo, e eles funcionam durante o programa (linhas 20 e 30).

* Pode-se ler e gravar registros sem ter que fechar e abrir o arquivo novamente (linhas 1090 e 2040).

* Não se pode usar as variáveis do comando FIELD em comandos LET ou INPUT. Usam-se outras variáveis para receber dados e estes são movidos usando RSET ou LSET (linhas 1020 e 1080).

CAPÍTULO III



INTRODUÇÃO AO HB-MCP

O HB-MCP é uma versão do CP/M desenvolvida especialmente para o seu HOTBIT. Portanto, tudo que voce já conhece sobre o CP/M valerá para o HB-MCP.

Como os comandos do CP/M são poucos e o HOTBIT tem um hardware arrojado, o HB-MCP, além de manter uma total compatibilidade com o CP/M, possui ainda várias vantagens, como veremos mais adiante.

Se você desejar mais informações sobre o CP/M em particular existem várias publicações técnicas a respeito do seu funcionamento interno, características, etc.

Neste capítulo vamos nos deter apenas no HB-MCP, seus comandos, utilitários e vantagens sobre o seu ancestral, o CP/M.

INICIALIZAÇÃO DO HB-MCP

Você recebeu junto com o seu drive dois disquetes que são muitos especiais, pois cada um deles contém um sistema operacional diferente, capaz de tornar seu HOTBIT muito mais versatil do que ele já é.

Um desses disquetes (veja a etiqueta) contém o sistema operacional HB-MCP e é este disquete que você deverá utilizar neste momento.

Veja no apêndice II como instalar o drive. Uma vez instalado e com o micro ligado (tendo, portanto, o DSK-BASIC em operação), insira o disco com o HB-MCP no drive A e produza um RESET no micro.

Ao ser dado RESET, o HOTBIT verifica se o drive contém algum disco com um novo sistema operacional gravado. Se tal sistema existir, será carregado para a memória e o contrôle do micro será transferido para ele.

Ao ser inicializado, o HB-MCP procura a existência de 64 Kbytes contínuos de memória RAM, que é a quantidade exata necessária para seu funcionamento. O seu HOTBIT já contém essa quantidade de memória e portanto você não precisa se preocupar com isso.

Após ter achado tal quantidade de memória, ele continua a pesquisa para ver se existem outros 64 Kbytes adicionais de RAM que funcionarão como pseudo-disco.

Estes 64 Kbytes a mais podem ser facilmente acoplados ao HOTBIT através do cartucho "expansão de memória", gerando assim um pseudo-drive com capacidade de 62 Kbytes de dados e uma grande velocidade de acesso. Isso pode ser muito útil quando usarmos programas que buscam constantemente dados em disco. Num disco real, o tempo de busca e leitura de um dado é muito maior que numa expansão de RAM.

Se existir um cartucho de 80 colunas instalado no micro, o HB-MCP o aciona automaticamente, permitindo assim o uso de 80 colunas no vídeo sem necessidade de se dar qualquer comando para por este cartucho em funcionamento.

Após toda essa "caça" aos cartuchos e expansões que

podem estar instalados no HOTBIT, o restante do HB-MCP é carregado e tranferido para o fim da memória, de acordo com a figura 3.2 .

A indicação de uma correta carga do sistema é percebida com o aparecimento da seguinte tela :

FIGURA 3.1 - Carga correta do HB-MCP.

```
HB-MCP   [EPCOM]   V1.0

A>
```

O tracejado piscante é chamado de CURSOR e indica em que local da tela aparecerá aquilo que você comandar.

ORGANIZAÇÃO INTERNA DA MEMÓRIA

Quando o HB-MCP é carregado para a memória do HOTBIT esta fica da seguinte maneira:

FIGURA 3.2 - O HB-MCP na memória do HOTBIT.

B I O S	&HFFFF &HE700	6,25 Kbytes
B D O S	&HE6FF &HD900	3,5 Kbytes
C C P	&HD8FF &HD100	2 Kbytes
T P A	&HD0FF &H0100	52 Kbytes
página base	&H00FF &H0000	0,25 Kbytes

No início da memória existem alguns pontos de entrada para as diversas funções do HB-MCP.

A região chamada TPA (Transient Program Area,ou Área de Programas Transientes) é onde são carregados os programas que você pede para serem executados.

O CCP é um programa interno do HB-MCP que serve de mediador entre o usuário (você) e o restante do HB-MCP. É ele quem recebe os comandos que você digita, como DIR, ERA, etc. É ele também quem carrega no TPA o programa que você pedir para ser executado, passando-lhe o contrôle.

O BDOS e o BIOS são os responsáveis por todo o funcionamento do HB-MCP. São eles quem controlam o acesso ao teclado, drives, vídeo, etc. É como se eles fossem o boy daquela nossa empresa imaginária que criamos no capítulo I . Quando você comanda DIR, por exemplo, o CCP pede para que o BDOS pesquise certas regiões estratégicas do drive. Recebendo as informações pesquisadas, o CCP seleciona as que interessam e pede para que o BDOS envie estes dados selecionados para o vídeo.

Como se pode ver, os programas apenas decidem o que fazer com os dados, mas é o BDOS (Basic Disk Operating System ou, Sistema Básico de Operação de Disco) e o BIOS (Basic INPUT/OUTPUT System ou, Sistema Básico de Entrada e Saída) quem busca e envia os dados aos periféricos.

Quando ocorre algum erro durante o acesso ao drive, como disco não inserido, drive mau regulado, etc, ele será detectado pelo BIOS e pelo BDOS antes de ser detectado pelo programa. Isso causa um inconveniente, porque dependendo do erro, o programa que estava sendo executado acaba sendo cancelado pelo próprio BDOS, recarregando o CCP e mostrando o sinal de "pronto" característico do HB-MCP: "A>".

Como o programa que estava sendo executado foi cancelado "sem mais nem menos", os dados que ele manipulava também foram ignorados, e a partir daí os arquivos que o programa utilizava acabaram ficando abertos (sem que o programa desse um comando CLOSE) e consequentemente não estão mais confiáveis para uso futuro. Daí a necessidade de se ter pelo menos uma cópia de reserva dos arquivos. Mais adiante esses erros serão descritos com mais detalhes.

Com essa subdivisão da memória, todos os programas são carregados no TPA. Uma vez carregados, quando quiserem acessar o teclado, o vídeo ou arquivos em disco, eles o fazem através do BDOS, acessando as tabelas no começo da memória que indicam onde estão o BDOS e o BIOS.

Dessa maneira, o programa que está rodando nem se interessa em como o BDOS fará o acesso desejado, só se importando com o resultado que receber. Assim, qualquer programa escrito para CP/M também rodará no HB-MCP. As diferenças do circuito eletrônico (hardware) do HOTBIT com outros micros não-MSX passam desapercebidas pelo programa. Apenas o BIOS é que sofre alteração de computador para computador, justamente porque é essa parte do CP/M que depende do hardware.No HOTBIT ela foi especialmente desenvolvida para aceitar todos os periféricos que este poderoso micro possa ter.

Apesar de ser totalmente compatível com o sistema operacional CP/M 2.2 , o HB-MCP é intrinsicamente diferente dele em todos os níveis.

O BDOS e o CCP apenas simulam o CP/M, possibilitando ao HOTBIT acesso aos softwares desenvolvidos para ele.

CCP - PROCESSADOR DE COMANDOS DO CONSOLE

No HB-MCP existe um pequeno "programa" que serve como mediador entre o operador (você) e o restante do sistema.

Este "programa" é na verdade uma parte integrante do HB-MCP e não um programa separado. Quando o HB-MCP está instalado e pronto para ser utilizado, quem tem o controle na realidade é o CCP.

O CCP existe para que você possa operar o computador dizendo qual programa você quer rodar, ou se deseja ver o diretório, etc.

Quando digitamos um comando, o CCP verifica se ele é reconhecível, e se for, executará sua função. Mas se não for um dos comandos que ele aceita, então irá procurar no disco algum programa que tenha como nome aquilo que você digitou, carregando-o para a memória e lhe passando o contrôle. Portanto, para se executar um programa, basta digitar seu nome.

O CCP envia para o terminal os símbolos "A>" para dizer que está pronto para receber um comando seu. A letra "A" significa que o seu comando surtirá efeito no drive A. Se você visualizar "B>", quer dizer que os comandos recebidos pelo CCP serão realizados no drive B e assim por diante.

FIGURA 3.3 - O sinal de "pronto" do CCP.

```
A>
```

Este "A>" é semelhante ao "Ok" do BASIC.

Quando então temos o "A>", podemos digitar um comando interno (um daqueles que o CCP aceita) ou o nome do programa que queremos utilizar.

Os comandos internos que veremos mais adiante só podem ser utilizados quando o CCP está rodando, ou seja, quando temos o "A>" no vídeo (ou B>, C>, etc).

Quando pedimos para executar um programa, este é carregado para a memória tomando o lugar do CCP, não sendo então possível utilizar mais tais comandos. Ao terminar a execução do programa pedido, o CCP é automaticamente recarregado para a memória, voltando a mostrar o "A>".

Todos os programas que são carregados na memória (na região chamada TPA) passam a gerenciar o funcionamento do micro com o auxílio do BDOS e do BIOS, não sendo mais necessário o uso do CCP. Afinal, apenas um "gerente" suficiente para comandar as ações executadas pelo micro.

Você já leu no capítulo II (DSK-BASIC) que alguns de seus comandos aceitam o ponto de interrogação (?) e o asterisco (*) para indicar ambiguidade nos nomes dos arquivos. Esses caracteres são conhecidos como "WILD CARDS" e são também aceitos pelo HB-MCP com o mesmo propósito que no DSK-BASIC.

A seguir vamos detalhar cada um dos comandos que o CCP aceita, como também a maneira de se executar os principais programas que acompanham o disco HB-MCP que você recebeu junto com o drive.

DIR

O comando DIR é um comando do CCP que mostra os arquivos presentes no drive especificado. Não sendo especificado nenhum drive, será assumido o drive corrente (aquele visualizado antes do ">").

FIGURA 3.4 - O comando DIR.

```
A>dir
A: PIP      COM : FORMAT   COM
A: DRINP    COM : STAT     COM
A: XSUB     COM : SUBMIT   COM
A: FUNCOES  COM : COPY     COM
A: DUMP     COM : COPDOS   COM
A: SYSGEN   COM : AUTOLD   COM
A: BACKUP   COM : TERM     COM
A: DSKCNV   COM : BATCH    COM
A: DROUT    COM
A>
```

Para especificar outro drive, basta indicá-lo logo após o comando DIR:

FIGURA 3.5 - Usando o DIR para o drive B .

```
A> DIR B:
Nenhum arquivo
A>
```

O comando DIR é semelhante ao FILES do DSK-BASIC, podendo-se usar, também, o asterisco (*) e o ponto de interrogação (?).

FIGURA 3.6 - Usando o ponto de interrogação (?) com o DIR.

```
A>
A>dir dr???.com
A: DRINP    COM : DROUT    COM
A>
```

Como todos os comandos do CCP fazem parte do HB-MCP, eles não são vistos pelo comando DIR. Este comando mostra apenas os arquivos gravados no disco e não os comandos internos do HB-MCP e é porisso que você não vê programas como DIR.COM , por exemplo. Mesmo que você grave um programa com este nome você não conseguirá executá-lo,pois ao digitar DIR o CCP irá mostrar o diretório do disco em vez de procurar um prorama com este nome. Sendo assim, não é possível executar programas que tenham como nome os comandos do CCP.

ERA

O comando ERA serve para apagar um arquivo do disco, de modo semelhante ao comando KILL do DSK-BASIC. Ele também admite o uso do asterisco (*) e do ponto de interrogação (?).

No caso de ser usado *.* indicando que todos os arquivos deverão ser apagados, uma pergunta lhe é feita para confirmar a operação.

O nome do arquivo a ser apagado deverá ser colocado logo após o comando ERA e a especificação do drive:

FIGURA 3.7 - Uso do comando ERA.

```
A>ERA B:LIXO.BAS
A>
```

No caso de não existir o arquivo, será mostrada uma mensagem de erro:

```
A>ERA ARQX
Nenhum arquivo
A>
```

ATENÇÃO: Não apague nenhum dos programas que estão no disco HB-MCP, pois estes são de grande importância. Mais adiante veremos como utilizar aguns deles e também como tirar copias reservas de todos para evitar perdas acidentais.

REN

O comando REN é utilizado quando queremos d um novo nome a um arquivo. Sua única utilização, portanto, é para mudar o nome de um arquivo que já existe no disco. Supondo que exista um arquivo de nome ARQ1 no disco do drive A e que queremos mudar seu nome para LIXO.BAS . Nesse caso, podemos utilizar o seguinte comando:

FIGURA 3.8 - Uso do comando REN .

```
A>REN LIXO.BAS=ARQ1
A>
```

Note que o arquivo não é alterado, mas apenas o seu nome é modificado.

TYPE

O comando TYPE exibe na tela o conteúdo de um arquivo. O arquivo deverá ser de texto, senão você verá coisas malucas no vídeo (mas não se preocupe que nada de mal deverá acontecer).

Quando temos um arquivo e queremos "ver" o que há gravado nele, devemos usar o comando TYPE.

O comando TYPE mostrará todo o arquivo até encontrar o caracter de código ASCII &H1A, que indica fim de arquivo.Se você tentar utilizar TYPE em um programa (os que tem .COM na extensão do nome) provavelmente obterá um show muito bonito

no seu vídeo, mas nada útil, já que um programa é um arquivo que só contém códigos de máquina e muitos deles são comandos de tela (posicionamento de cursor, etc).

Para se "congelar" a listagem durante a execução do TYPE, tecle ^S (ou STOP) que o vídeo ficará parado até ser teclado novamente ^S (ou STOP), que "descongelará" o vídeo. Se você digitar qualquer outra tecla além de ^S, o vídeo também será descongelado. Mais adiante serão vistos o ^S e outros "CONTROL s" que produzem efeito no HB-MCP.

SAVE

O comando SAVE grava um arquivo em disco como uma imagem da memória. Este comando tem muito pouca utilidade e provavelmente você nunca o utilizará. Mesmo assim, vale a pena saber como usá-lo.

Para utilizá-lo, é necessário dizer quantas "páginas" de memória deverão ser gravadas, já que cada página contém 256 bytes de tamanho. O endereço inicial da memória que é gravado é o &H0100. A partir dele são carregados todos os programas que rodam no HB-MCP. Se, por exemplo, tivermos um programa de 2 Kbytes de tamanho em disco, após ele ter sido executado e retornado o controle para o CCP, ele ainda permanecerá na memória, pois o CCP ocupa uma região dela onde praticamente permanece sozinho. Nessa situação, podemos dar o seguinte comando.

FIGURA 3.9 - Uso do comando SAVE .

```
A>SAVE 8 COPIA.XXX
A>
```

Com este comando, será criado um arquivo em disco contendo 8 páginas de memória (8x256=2K) de nome COPIA.XXX e que é idêntico aquele programa que tinha acabado de ser executado. Na realidade, simplesmente tiramos uma cópia daquele programa.

Existe um programa próprio para se fazer cópias de arquivos e que é muito melhor e mais confiável que o comando SAVE, tornando-o praticamente inútil. Nós o veremos mais adiante.

USER

O comando USER é outro raramente utilizado.

Ele indica qual o usuário que está utilizando o computador. Todo acesso a qualquer arquivo em disco só é feito após uma comparação do usuário atual com o usuário que criou o arquivo. O HB-MCP só libera o arquivo para o uso se houver coincidência na comparação.

Quando o HB-MCP é inicializado, é assumido o USER 0.

Para todos os arquivos que forem criados será assu-

mido o USER 0. Se, por exemplo, você modificar o USER para USER 5, todos os arquivos do USER 0 não estarão mais disponíveis. Para que não haja um "carnaval" de USER's no seu disco, é mais prático e recomendável utilizar apenas o USER 0, que é assumido quando o sistema é carregado. Mas, se mesmo assim você quiser se divertir um pouco, pode-se mudar o USER com o seguinte comando:

FIGURA 3.10 - Uso do comando USER .

```
A>USER n
```

O valor de n pode ir de 0 a 15. Para se voltar ao USER 0 basta o seguinte comando:

```
A>USER 0
A>
```

Quando estamos utilizando um USER diferente de 0, isso é indicado antes do "A>":

```
5A>
```

Um uso do comando USER pode ocorrer quando existir, acoplado ao seu HOTBIT, um disco com enorme capacidade de armazenamento (por exemplo, com uns 20 Mbytes=20.000.000 bytes).

Nesse disco, por caberem muitos arquivos e programas, é uma boa prática separá-los por grupos com o auxílio do comando USER: USER 1 para os programas e arquivos da folha de pagamento; USER 2 para os programas e arquivos da contabilidade; USER 3 para os jogos que jocê mais gosta, e assim por diante. Com uma separação assim, fica fácil utilizar um certo sistema, pois só deixamos disponíveis os programas e arquivos que nos interessam em específico. Imagine a dificuldade em achar um certo jogo num disco com 20 MBytes contendo muitos programas e arquivos! Mesmo com o auxílio do ponto de interrogação (?) e do asterisco(*), seria penosa a procura.

O inconveniente deste método de separação em grupos é que alguns programas são muito utilizados, sendo sua disponibilidade quase que obrigatória, como o programa PIP.COM, por exemplo.

Separando os arquivos e programas com o comando USER teríamos que ter uma cópia do PIP.COM e de mais alguns outros comandos de uso frequente em cada separação de USER que tivermos em uso naquele disco, gerando assim um "repeteco" de programas que é um desperdício.

Como o disco que você tem no seu HOTBIT armazena por volta de 340 Kbytes, o USER acaba sendo não muito útil.

Embora possa operar com apenas um drive, o HB-MCP foi desenvolvido tendo em vista o uso de dois drives, sendo esta a melhor configuração a ser empregada.

Supondo que o drive B esteja conectado, podemos tê-lo como drive corrente, com o seguinte comando:

```
A>B:
B>
```

Quando temos no vídeo B> em vez de A>, significa que os comandos que forem dados sem ser especificado um drive, serão executados no drive B. Para voltar a operar diretamente com o drive A, basta o seguinte comando :

```
B>A:
A>
```

Com os comandos do HB-MCP, não é possível fazer cópias de arquivos, nem saber quanto espaço há ainda livre no disco. Para resolver este inconveniente, existem vários programas feitos especialmente para cópias de arquivos, para mostrar a ocupação do disco etc.

No HB-MCP os programas devem possuir a extensão .COM no nome para que o CCP os reconheça como programa, e não como um arquivo qualquer.

Ao ser digitado um comando que não seja do próprio HB-MCP (DIR, ERA, TYPE, SAVE, REN e USER) o CCP coloca a extensão .COM e procura no disco algum programa com este nome (por isso não é necessário você digitar .COM quando quiser rodar algum programa, pois o próprio CCP já faz isso).

Você já deve ter reparado que todos os arquivos que vieram com o disco HB-MCP possuem a extensão ".COM" sendo, portanto, programas executáveis.

Cada um desses programas tem uma finalidade específica e vamos mostrar como utilizar alguns deles que são também conhecidos como comandos externos, dada a grande frequência com que são utilizados.

Todos os comandos que vimos até aqui, podem ser dados em uma única linha, desde que separados por vírgula (,).

Vamos supor que temos dois drivers instalados e que queremos saber o que está gravado nos disco lá presentes. Podemos então comandar DIR e teremos todos os arquivos gravados no disco do drive A. Logo após, comandamos DIR B: e teremos mostrados todos os arquivos gravados no disco do drive B. Uma outra forma de obter o mesmo resultadoé comandar:

```
DIR , DIR B:
```

Assim, serão mostrados os arquivos no disco do drive A e, logo em seguida, serão mostrados os arquivos presentes no disco do drive B.

Com o tempo, quando você já estiver bem familiarizado com o HB-MCP, verá que é comum usarmos mais de um comando para obtermos o resultado desejado e isto fica facilitado com a possibilidade de usarmos a vírgula para separarmos tais comandos.

FORMATAÇÃO DE DISQUETES NO HB-MCP & CóPIA DO SISTEMA OPERACIONAL

Você já deve estar careca de saber quando e porque devemos formatar um disco.

Para se formatar um disco no HB-MCP, é necessário um programa específico, pois mesmo sendo um sistema operacional próprio para o HOTBIT, ele é totalmente independente do cartucho controlador do drive. Porisso existe um programa chamado FORMAT.COM que serve para formatar os discos, deixando-os prontos para serem usados pelo HB-MCP.

A formatação empregada no HB-MCP é diferente daquela utilizada pelo DSK-BASIC e pelo HB-DOS (que será detalhado no próximo capítulo).

Sendo assim, os discos usados no HB-MCP deverão ser usados apenas nele e se você tentar utilizá-los no DSK-BASIC ou no HB-DOS provavelmente não conseguirá.

Para se formatar um disco no HB-MCP, deve-se executar o programa FORMAT.COM e logo em seguida indicar em qual drive será colocado o disco a ser formatado, por exemplo:

```
A>FORMAT A:
```

Com este comando, o programa FORMAT.COM é carregado na memória e é executado, mostrando a seguinte tela :

FIGURA 3.11 - Primeira tela do comando FORMAT .

```
FORMAT
Inicia Disquetes
V1.0

Coloque disquete para ser
formatado no dirve A:

Tecle <RETURN>
```

Neste instante, retire o disco que está no drive A e insira o disco a ser formatado, teclando RETURN logo em seguida, sendo então visualizada a seguinte tela :

FIGURA 3.12 - Segunda tela do comando FORMAT .

```
FORMAT
Inicia Disquetes
V1.0


Coloque disquete para ser
formatado no dirve A:

Tecle <RETURN>

Confirma (S/N) ?
```

Cuidado para não formatar o disco com o HB-MCP !!!

Tenha certeza de que o disco que está no drive A é realmente o disco correto, teclando então a letra S para se iniciar o processo de formatação ou a letra N para se interromper o programa FORMAT.COM.

Após teclar S, é iniciada a formatação do disco, sendo mostrada qual trilha está sendo formatada.

Terminada a formatação e se esta estiver ocorrido normalmente é perguntado se deseja fazer uma verificação.

Sempre é bom verificar se a formatação está correta ou não, pois é uma garantia a mais de que o disco está pronto para uso. Tecle então S novamente para iniciar a verificação.

FIGURA 3.13 - Terceira tela do programa FORMAT .

```
FORMAT
Inicia Disquetes
V1.0

Coloque disquete para ser
formatado no dirve A:

Tecle (RETURN)

Confirma (S/N) ? S

Formatando trilha 079

Formatação completa

Verifica (S/N) ?
```

Após a verificação (se esta foi executada) é perguntado se deseja formatar algum outro disquete, podendo ser formatados vários disquetes de uma só vez sem ter que chamar o programa FORMAT.COM para cada formatação a ser feita. Ao terminar de formatar o último disquete, responda não à esta pergunta teclando a letra N (figura 3.14).

Você está 2 vezes careca de saber que um disquete após ter sido formatado está completamente limpo, como é o caso do disco que acabou de ser formatado no drive A.

Assim, o programa FORMAT.COM pede para que seja colocado neste drive um disco que contenha o sistema operacional HB-MCP para que possa ser recarregado então o CCP. Retire o disco presente no dirve A e coloque o disco HB-MCP ou algum outro que contenha este sistema operacional gravado nele, teclando RETURN logo em sequida.

Se você possuir dois drives, pode efetuar a formatação do disco no drive B em vez de efetuá-la no drive A com o seguinte comando :

```
A>FORMAT B:
```

FIGURA 3.14 - Quarta tela do comando FORMAT .

```
FORMAT
Inicia Disquetes
V1.0


Coloque disquete para ser
formatado no dirve A:

Tecle <RETURN>

Confirma (S/N) ? S

Formatando trilha 079

Formatação completa

Verifica (S/N) ? S

Verificando trilha 079

Verificação completa

Formata outro (S/N) ? N

Coloque disquete com sistema
Tecle <RETURN>
```

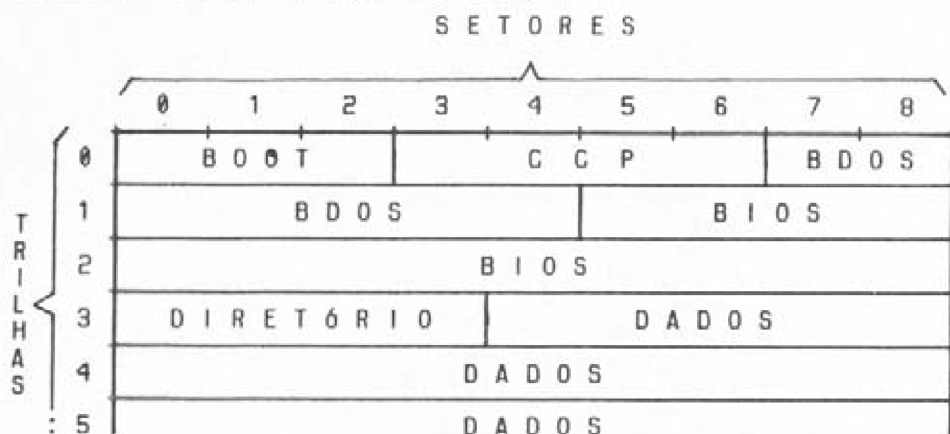
Se o disco a ser formatado possuir algum defeito, o programa FORMAT.COM não conseguirá formatá-lo corretamente e emitirá uma mensagem de erro. Se isso ocorrer, é melhor se desfazer deste disco e utilizar um outro, pois se não é possível formatar um disco, com certeza não será possível utilizá-lo posteriormente.

Após a formatação de um disco no HB-MCP é aconselhável fazer uma cópia do sistema operacional para ele. Toda vez que um programa termina de ser executado, o CCP ,que faz parte do HB-MCP, é carregado para a memória. Se nesse disco não existir uma cópia do CCP, após a execução de um programa o micro acabará ficando "maluco", pois tentará carregar o CCP para a memória e nada irá encontrar. Por isso é necessário existir uma cópia do sistema operacional HB-MCP em cada disco que for utilizado.

O HB-MCP está gravado nas três trilhas iniciais, a trilha 0, 1 e a trilha 2 . Estas três trilhas do disco são sempre reservadas para o HB-MCP (figura 3.15).

Para se tirar uma cópia do sistema operacional, basta copiar estas trilhas de um disquete que já contenha o HB-MCP para o disquete que foi formatado. E o programa SYSGEN.COM serve exatamente para isso.

FIGURA 3.15 - Trilhas reservadas para o HB-MCP.



Todo disquete novo que for formatado no HB-MCP terá que receber o sistema operacional logo em seguida através do programa SYSGEN.COM. Para fazer isso, coloque o disquete que você recebeu com o HB-MCP e digite o seguinte comando :

A>SYSGEN (e tecle RETURN)

A partir daí, o programa SYSGEN lê as trilhas 0, 1 e 2 do disco que esta no drive A e as carrega para a memória. Após a leitura, ele pede para ser colocado no drive A o disco que receberá o sistema operacional, esperando então ser teclado RETURN.

FIGURA 3.16 - Trocando o disco para receber o sistema.

```
SYSGEN
Gera Sistema
V1.0


Coloque disquete no drive A:
para receber sistema

Tecle <RETURN>
```

Retire então o disco do drive A e coloque o disco que acabou de ser formatado, teclando então RETURN.

Passados alguns segundos, o SYSGEN terá gravado nas trilhas 0, 1 e 2 do disco uma cópia do HB-MCP que foi lido do disco original, assim quando você ligar o micro com este novo disco no drive, você entrará no HB-MCP normalmente.

Após a cópia ter sido efetuada sem problemas, o programa pergunta se o usuário deseja fazer uma outra cópia para um outro disquete permitindo assim várias cópias em vários discos a partir de um mesmo original.

FIGURA 3.17 - Opção de cópias em vários discos.

```
SYSGEN
Gera Sistema
V1.0


Coloque disquete no drive A:
para receber sistema

Tecle <RETURN>

Outra cópia (S/N) ?
```

Se você responder S, então é pedido para ser colocado o disco que receberá a cópia no drive A e o processo se repete.

Ao responder N, é pedido o disco que contém o sistema operacional HB-MCP gravado no drive A.

Note que como o disco que está no drive A agora acabou de receber o sistema operacional, você pode teclar RETURN que você terá o A> sem trocar o disco.

Não se esqueça porém que apenas o sistema operacional é que foi gravado !!! Os programas e arquivos que estão no disco original não foram gravados. Para fazer uma cópia destes programas é necessário utilizar um programa de cópias de arquivos e não um programa de cópia de sistema operacional como é o caso do SYSGEN.COM.

FIGURA 3.18 - Finalização do SYSGEN .

```
SYSGEN
Gera Sistema
V1.0


Coloque disquete no drive A:
para receber sistema

Tecle <RETURN>

Outra cópia (S/N) ? N

Coloque disquete com sistema
Tecle <RETURN>
```

Se durante o processo de cópia do sistema operacional ocorrer algum erro, o programa SYSGEN.COM emitirá uma mensagem de erro.

FIGURA 3.19 - Erro durante a cópia do sistema.

```
SYSGEN
Gera Sistema
V1.0


Coloque disquete no drive A:
para receber sistema

Tecle <RETURN>

Erro na gravação !

Coloque disquete com sistema
Tecle <RETURN>
```

Quando isso ocorrer, tente reformatar o disco e recopiar o HB-MCP. Se mesmo assim o erro persistir, é melhor utilizar outro disquete e se desfazer desse.

Para se copiar os programas e arquivos, você poderá utilizar o programa PIP.COM se você tiver dois drives (A e B) ou o programa COPY.COM se você só tiver um drive, e estes programas estão descritos logo mais adiante.

STAT.COM

O programa STAT.COM foi desenvolvido com o objetivo de mostrar informações sobre o disco ou sobre os arquivos.

Com esse programa podemos saber, por exemplo,o quanto ainda temos de espaço livre num disco.

Para isso, basta usar o seguinte comando:

```
A>STAT
```

O vídeo deverá ficar assim:

FIGURA 3.20 - Espaço livre em disco através do STAT.

```
STAT
Informações sobre o disco
V1.0

A: R/W, Espaço: 288k
```

Por não ser um de seus comandos internos, o CCP procurou no drive corrente (no caso, o drive A) um programa de nome STAT.COM . Sendo encontrado, ele foi carregado para a memória do micro e passou a ser executado.

O programa STAT, ao ser iniciado sem receber nenhum parâmetro, simplesmente verificou no drive o quanto havia de espaço disponível para uso (288 Kbytes). Mostrou, ainda, qual o drive que foi examinado (A:) e que o disco lá presente está no modo R/W. Isso quer dizer que neste disco podem ser lidos (READ) e gravados (WRITE) dados.

Se você possuir dois drives, pode ser utilizado o seguinte comando para se saber o espaço disponível no disco instalado no drive B:

```
A>STAT B:
```

Observe que no exemplo anterior foi especificado o parâmetro B: no comando STAT.

Através do programa STAT.COM podemos saber também o tamanho de um arquivo. Se for dado como parâmetro o nome de um arquivo, o programa STAT.COM irá pesquisá-lo no disco indicado e mostrará algumas informações muito úteis sobre ele. Veja, a seguir, um exemplo:

```
A>STAT STAT.COM
```

O resultado desse comando é o seguinte:

FIGURA 3.21 - Dados do STAT.COM obtidos com ele mesmo.

```
STAT
Informações sobre o disco
V1.0


 Regs  Bytes  Ext Atr
   40     6k    1 R/W A:STAT.COM
Bytes restantes em A: 288k
```

Usou-se como parâmetro o próprio programa STAT.COM . Veja a seguir a interpretação de algumas informações fornecidas pelo programa:

* O item "Regs" (de registros) significa quantos registros de 128 bytes o programa STAT.COM ocupa (40 , no exemplo anterior).

* O item "Bytes" mostra o tamanho do arquivo. Se você conferir,40 registros com 128 bytes cada resulta num total de 5120 bytes. Isso significa que o programa STAT.COM tem realmente 5120 e não os 6 Kbytes mostrados no item "Bytes". Só não se esqueça que o disco é como um livro e o espaço nele alocado é em blocos (semelhante às páginas do livro). No HB-MCP, cada bloco de disco ocupa 2 Kbytes e portanto, todo arquivo gravado ocupará sempre algum múltiplo de 2 Kbytes. Conclusão: o programa STAT.COM tem realmente 5120 bytes, mas no disco ocupa 6 Kbytes.

* O item "Ext" (de extensão) mostra quantos "pedaços" do diretório o programa STAT.COM utiliza. O diretório do disco contém várias extensões (pedaços) e cada extensão tem espaço suficiente para gerenciar 32 Kbytes de disco. Se um arquivo ocupa menos do que 32 Kbytes, então com apenas uma extensão do diretório já é possível saber exatamente como os bytes do arquivo foram gravados no disco. Mas se ele possuir,por exem-

plo, 40 Kbytes de tamanho, então uma só extensão do diretório não será suficiente. Serão necessárias duas estensões do diretório: uma para gerenciar os primeiros 32 Kbytes do arquivo e a outra para gerenciar os 8 Kbytes restantes. Como o programa STAT.COM ocupa apenas 6 Kbytes do disco, então uma só extensão do diretório já é suficiente.

* O item "Atr" (de atributo) mostra o atributo do arquivo. Este arquivo pode ser R/W (de READ/WRITE) ou R/O (de READ/ONLY). Isso, na verdade, é uma proteção contra alterações indevidas. Se um arquivo tem atributo R/W, significa que podemos alterar os dados lá gravados sem nenhum problema, inclusive apagar o arquivo (com o comando ERA, por exemplo). Mas se o arquivo tem atributo R/O, ele só poderá ser lido, criando assim uma proteção aos dados do arquivo e também do próprio arquivo.

O programa STAT.COM também aceita os "Wild Cards" (* e ?) no nome do arquivo dado como parâmetro. Se você quiser ver o tamanho e atributo de cada arquivo gravado no disco, dê o seguinte comando:

```
A>STAT  *.*
```

O vídeo ficará assim:

FIGURA 3.22 - Uso do asterisco (*) com o STAT.COM .

```
V1.0

 Regs  Bytes  Ext Atr
    4     2k    1 R/W  A:AUTOLD.COM
    8     2k    1 R/W  A:BACKUP.COM
    4     2k    1 R/W  A:BATCH.COM
   95    12k    1 R/W  A:COPDOS.COM
   17     4k    1 R/W  A:COPY.COM
    8     2k    1 R/W  A:DRINP.COM
    8     2k    1 R/W  A:DROUT.COM
    8     2k    1 R/W  A:DSKCNV.COM
    5     2k    1 R/W  A:DUMP.COM
   11     2k    1 R/W  A:FORMAT.COM
    2     2k    1 R/W  A:FUNCOES.COM
   59     8k    1 R/W  A:PIP.COM
   40     6k    1 R/W  A:STAT.COM
    9     2k    1 R/W  A:SUBMIT.COM
    5     2k    1 R/W  A:SYSGEN.COM
    2     2k    1 R/W  A:TERM.COM
    6     2k    1 R/W  A:XSUB.COM
Bytes restantes em A: 288k

A>
```

Repare que os arquivos são mostrados em ordem alfabética, o que facilita muito a procura de um arquivo em espe-

cífico. Mas em compensação, se você um dia possuir um disco com muitos arquivos gravados e utilizar *.* como parâmetro do programa STAT, fatalmente terá que aguardar vários segundos até que o STAT.COM os coloque em ordem alfabética antes de apresentá-los no vídeo.

A esta altura você já deve estar se perguntando como fazer para modificar o atributo de um arquivo. Você pode modificá-lo quando bem entender, com o auxílio do programa STAT.COM. Quando um arquivo ou um programa está assinalado com R/O, pode apenas ser lido, porém, não pode ser modificado. Isso é muito útil em programas que não devem ser apagados do disco, ou mesmo em arquivos importantes que não podem ser "acidentalmente" modificados. Se um arquivo ou um programa não está como R/O, ele está como R/W, pois só existem esses dois estados.

Usando o programa STAT.COM pode-se assinalar com R/W ou R/O cada arquivo ou programa, em separado ou em grupos, através do ponto de interrogação (?) e do asterisco (*).

O símbolo de cifrão ($) é utilizado para indicar ao STAT que se deseja modificar a situação para R/O ou para R/W.

Você pode proteger todos os programas em um disco com o seguinte comando:

```
A>stat *.com $r/o
```

A tela deve ficar assim:

FIGURA 3.23 - Protegendo todos os programas com o STAT.COM .

```
STAT
Informações sobre o disco
V1.0

PIP.COM alterado para R/O
FORMAT.COM alterado para R/O
DRINP.COM alterado para R/O
STAT.COM alterado para R/O
XSUB.COM alterado para R/O
SUBMIT.COM alterado para R/O
FUNCOES.COM alterado para R/O
COPY.COM alterado para R/O
DUMP.COM alterado para R/O
COPDOS.COM alterado para R/O
SYSGEN.COM alterado para R/O
AUTOLD.COM alterado para R/O
BACKUP.COM alterado para R/O
TERM.COM alterado para R/O
DSKCNV.COM alterado para R/O
BATCH.COM alterado para R/O
DROUT.COM alterado para R/O
A>
```

Observe no exemplo anterior que o programa STAT foi mostrando cada arquivo e programa que estava sendo assinalado como R/O. Observe também que não é mais possível apagar com o comando ERA os arquivos e programas assinalados como R/O.

Se você realmente desejar apagar um arquivo ou programa R/O, terá primeiro que passá-lo para R/W (através do programa STAT) e só depois poderá apagá-lo.

FIGURA 3.24 - Procedimento para apagar um arquivo R/O .

```
A>stat arq1.txt $r/w


STAT
Informações sobre o disco
V1.0


ARQ1.TXT alterado para R/W
A>
A>era arq1.txt
A>
```

Além dos indicadores R/W e R/O, existe um outro indicador nos arquivos e programas para selecioná-los como arquivos de sistema (ou programas do sistema).

Um arquivo ou programa pode ser do sistema ou não. Se ele for do sistema, seu nome não aparecerá quando for dado o comando DIR, embora ele esteja presente no disco.

Na verdade, este indicador de sistema seria melhor chamado de indicador do comando DIR, pois sua única função é inibir a atuação do comando DIR em alguns arquivos.

As vezes, quando temos muitos arquivos e programas em um disco, é conveniente "escondermos" alguns deles do comando DIR para facilitar a leitura, e é por isso que existe o indicador de sistema.

O programa STAT.COM é quem posiciona este indicador, de forma análoga ao indicador R/O e R/W visto anteriormente.

São usadas as palavras DIR e SYS logo após o símbolo do cifrão ($) para indicar se aquele arquivo ou programa será mostrado pelo comando DIR ou não.

Como exemplo vamos passar o programa FORMAT.COM para programa de sistema. Observe a figura 3.25 . Ela ilustra o procedimento necessário para isso.

Note que nesse exemplo, imediatamente após o programa FORMAT.COM ter sido assinalado como de sistema, seu nome não aparecerá mais no comando DIR.

Pode-se usar também o ponto de interrogação (?) e o asterisco (*) para se assinalar vários programas e arquivos com apenas um comando.

Observe também que, quando utilizamos STAT *.* , os arquivos e programas que não são mostrados pelo comando DIR estão com seus nomes entre parenteses ( ) indicando que são arquivos de sistema (figura 3.26).

FIGURA 3.25 - FORMAT.COM como programa de sistema.

```
A>stat format.com $sys



STAT
Informações sobre o disco
V1.0

FORMAT.COM alterado para SYS
A>
```

FIGURA 3.26 - Vendo os arquivos de sistema com o STAT.COM .

```
A>STAT *.*


V1.0

 Regs  Bytes  Ext Atr
    4     2k    1 R/W A:AUTOLD.COM
    8     2k    1 R/W A:BACKUP.COM
    4     2k    1 R/W A:BATCH.COM
   95    12k    1 R/W A:COPDOS.COM
   17     4k    1 R/W A:COPY.COM
    8     2k    1 R/W A:DRINP.COM
    8     2k    1 R/W A:DROUT.COM
    8     2k    1 R/W A:DSKCNV.COM
    5     2k    1 R/W A:DUMP.COM
   11     2k    1 R/W A:(FORMAT.COM)
    2     2k    1 R/W A:FUNCOES.COM
   59     8k    1 R/W A:PIP.COM
   40     6k    1 R/W A:STAT.COM
    9     2k    1 R/W A:SUBMIT.COM
    5     2k    1 R/W A:SYSGEN.COM
    2     2k    1 R/W A:TERM.COM
    6     2k    1 R/W A:XSUB.COM
Bytes restantes em A: 288k

A>
```

Fique atento, pois todos os outros comandos atuam sobre os arquivos e programas do sistema. Apenas o comando DIR é que os ignora. Se você comandar ERA em um arquivo ou programa de sistema, este será apagado do disco normalmente.

No HB-MCP, existe a possibilidade de se direcionar os periféricos de várias maneiras distintas.

Temos no HB-MCP quatro dispositivos lógicos:

1- Console (CON: para entrada e saída)
2- Entrada de sinal (ENS: para entrada, apenas)
3- Saída serial (SDS: para saída, apenas)
4- Impressora (LST: para saída, apenas)

Todos os programas feitos para o HB-MCP podem enviar ou receber dados de um desses 4 periféricos. Mas o seu HOT-BIT é muito flexível, e pode ter mais periféricos do que os descritos anteriormente. Por exemplo: interface RS-232, vídeo de 80 colunas, etc.

O programa STAT.COM além das funções já vistas, pode ser usado para associar a cada um desses periféricos lógicos um periférico físico.

Assim que o HB-MCP é carregado para a memória, é assumida a seguinte configuração:

Console (CON) = teclado e vídeo de 40 colunas
Impressora (LST) = impressora paralela

Se você possuir um cartucho de 80 colunas ligado ao seu HOTBIT, ele será automaticamente acionado no lugar do vídeo de 40 colunas.

Estas associações entre dispositivos lógicos e dispositivos físicos funcionam da seguinte maneira: quando um programa pede um dado através do teclado, ele o faz através do BDOS e do BIOS. Este dado pode ser fornecido pelo teclado realmente ou por um outro dispositivo acoplado ao HOTBIT, como por exemplo a interface RS-232. Como resultado, o programa "pensa" que está recebendo dados do teclado, quando na verdade os dados estão sendo enviados por um outro dispositivo. Isso dá uma enorme flexibilidade ao equipamento.

Veja a seguir o que pode ser associado a cada um dos periféricos lógicos:

CONSOLE (CON:) {
- teclado + vídeo de 80 colunas (TV8:)
- teclado + vídeo de 40 colunas (TV4:)
- dispositivos lógicos, entrada serial + impressora
- dispositivo impressora

ENTRADA SERIAL (ENS:) {
- entrada serial
- teclado

SAÍDA SERIAL (SDS:) {
- saída serial
- saída serial com eco, mostrado no no dispositivo console

IMPRESSORA (LST:) { vídeo de 80 colunas (TV8:)
vídeo de 40 colunas (TV4:)
impressora paralela
saída serial }

Com o programa STAT.COM, podemos fazer as devidas associações indicando os dispositivos lógicos e físicos.

Se você tiver um cartucho 80 colunas e quiser configurar a console (CON:) para ele, dê o seguinte comando:

```
A>STAT CON:=TV8:
```

Com ele, todos os dados que seriam enviados ao vídeo serão transferidos ao cartucho de 80 colunas.

Se você desejar ter no vídeo tudo que sairia na impressora, basta digitar:

```
A>STAT LST:=VD4:        (para 40 colunas)   ou
A>STAT LST:=VD8:        (para 80 colunas)
```

Para saber a configuração atual, use o seguinte comando:

```
A>STAT DEV:
```

O vídeo mostrará algo assim:

FIGURA 3.27 - Configuração atual com o STAT.COM .

```
STAT
Informações sobre o disco
V1.0


CON: é TV4:

ENS: é ESR:

SDS: é SSR:

LST: é IMP:

A>
```

Além de todas estas funções executadas pelo programa STAT.COM, ele ainda nos mostra algumas características do equipamento em sí, mais especificamente sobre o drive.

Você pode saber as características do drive usando o programa STAT.COM da seguinte maneira:

```
A>STAT DSK:
```

Em resposta, ele lhe dará informações como as apresentadas na figura 3.28 .

FIGURA 3.28 - Resultado do DSK: como STAT num drive de 3,5".

```
STAT
Informações sobre o disco
V1.0


       A: Características do disco
    2768: Registros de 128 bytes
     346: Capacidade em kbytes
      64: Entradas no diretório
      64: Entradas verificadas
     256: Registros por entrada
      16: Registros por bloco
      36: Registros por trilha
       3: Trilhas reservadas

A>
```

Este comando nos fornece alguns dados técnicos que fogem um pouco do objetivo desse livro, por isso não vamos nos deter muito nessa função do programa STAT. Além do que uma boa análise nos dados por ela apresentados já é o bastante para se ter uma idéia a respeito de seu uso.

Caso você se esqueça de tudo que o comando STAT.COM pode fazer, dê o seguinte comando:

```
A> STAT  VAL:
```

O vídeo apresentará um resumo das opções de uso do STAT.COM .

FIGURA 3.29 - Resultado do STAT com o parâmetro VAL: .

```
Informações sobre o disco
V1.0



Dados dos arquivos: d:arquivo.tipo $S
    ($S opcional p/ tamanho)
Disco temporariamente R/O: d:=R/O
Atributos: d:arquivo.tipo $R/O $R/W $SYS
 $DIR
Dados do disco: DSK: ou d:DSK:
Usuários: USR:

Dispositivos ativos: DEV:

Mudança de dispositivos:
 lógico=físico,lógico=físico,...

Dispositivos lógicos e físicos:
CON: = TV8: TV4: BAT: TSS:
ENS: = ESR: TCL: ES1: ES2:
SDS: = SSR: SSE: SS1: SS2:
LST: = VD8: VD4: IMP: LSR:
A>
```

Como você já percebeu, o programa STAT.COM é muito poderoso e nos fornece valiosas informações sobre o disco, periféricos e arquivos, sendo sua presença quase que obrigatória nos vários disquetes que você venha a possuir no futuro.

PIP.COM

O programa PIP.COM (Programa para Intercâmbio entre Periféricos) existe para possibilitar a cópia de arquivos de um periférico para outro. Com o PIP.COM podemos enviar e receber dados de qualquer periférico, mas seu uso mais comum é copiar arquivos de um disco para outro apenas.

Quando comandamos PIP, esse programa é carregado para a memória e indica através de um asterisco (*) que está pronto para receber um comando.

```
PIP
Intercâmbio de Periféricos
V1.0

*
```

Para se copiar um arquivo de um disco para outro, devemos indicar primeiramente qual o nome do arquivo que receberá a cópia e seu respectivo drive, depois devemos digitar o sinal de igual (=) e logo em seguida devemos indicar qual o nome do arquivo que será copiado.

Se quisermos copiar o próprio programa PIP.COM que está no drive A para o drive B, devemos dar o seguinte comando:

```
*b:pip.com=a:pip.com
```

Após ter sido efetuada a cópia, o programa PIP.COM mostra o asterisco novamente à espera de um novo comando. Para terminar a execução do PIP, devemos teclar ^C.

```
*b:pip.com=a:pip.com

*^C
A>
```

Quando o arquivo que receberá a cópia tiver o mesmo nome do arquivo a ser copiado nem precisamos dar o nome do primeiro arquivo ; o PIP assumirá o nome do arquivo original.

Sendo assim, para copiarmos o programa PIP.COM do drive A para o drive B, podemos dar o seguinte comando:

```
*b:=a:pip.com

*^C
A>
```

Com este comando, o programa PIP.COM que estava no

drive A foi copiado com o mesmo nome para o drive B.

Você já deve ter reparado que os exemplos apresentados até aqui sobre o programa PIP.COM foram o de fazer cópias de arquivos de um disco para outro. Pode-se porém serem efetuadas cópias de arquivos para um mesmo disco através do programa PIP.COM, só que será necessário dar um novo nome ao arquivo que receberá a cópia já que não tem muito sentido copiar um arquivo para ele mesmo.

Vamos supor que temos no drive A um arquivo de nome ARQ1.TXT e queremos fazer uma cópia deste arquivo para o drive A mesmo. Neste caso devemos escolher um nome apropriado para a cópia, como por exemplo ARQ1.COP. Depois podemos dar o seguinte comando :

```
*a:arq1.cop=a:arq1.txt

*^C
A>
```

Observe que agora temos duas vezes o mesmo arquivo no drive A, só que com nomes diferentes.

Se voce possui apenas um drive, não conseguirá através do programa PIP.COM efetuar cópias de arquivos de um disco para outro.

O programa PIP.COM não foi estruturado para permitir troca de disco durante o processo de cópia, e para se conseguir copiar arquivos de um disco a outro possuindo apenas um drive, deve-se utilizar o programa COPY.COM que foi desenvolvido justamente para isso.

O programa PIP.COM aceita também o ponto de interrogação (?) e o asterisco (*) para copiar mais de um arquivo com apenas um só comando. Quando ou o ponto de interrogação ou o asterisco são utilizados, o PIP vai mostrando quais arquivos estão sendo copiados.

Para tirar uma cópia de todos os programas que estiverem no disco HB-MCP, devemos dar o seguinte comando:

```
*B:=A:*.*
```

A medida que os arquivos vão sendo copiados, o PIP vai mostrando seus respectivos nomes.

```
*b:=a:*.*

Copiando...

PIP.COM

   .
   .
   .
*^C
A>
```

Podemos dar o comando para o PIP logo após seu nome. Quando fazemos isso, o PIP faz a cópia e retorna para o sistema operacional sem a necessidade de teclarmos ^C.

```
A>PIP B:=A:PIP.COM
A>
```

Quando copiamos um arquivo com o programa PIP.COM, este começa a fazer a cópia em um arquivo temporário de trabalho que possui a extensão .$$$ no nome.

Vamos supor que temos no drive B um arquivo chamado ARQ1 e que temos no drive A outro arquivo chamado ARQ1 também só que uma verão mais atual.

Se usamos o programa PIP.COM para copiar o arquivo ARQ1 do drive A para o drive B com o proprósito de atualização, ocorrerá o seguinte :

* Será feita uma cópia do arquivo ARQ1 do drive A para o ar-arquivo ARQ1.$$$ no drive B.
* Após a cópia ter sido efetuada sem problemas, o PIP apagará do drive B o arquivo ARQ1 caso este exista.
* Será mudado o nome do arquivo ARQ1.$$$ para ARQ1.

Com este método, voce não corre o risco de perder algum arquivo durante o processo de cópia mesmo que acabe a força ou ocorra algum problema no drive ou no disco. O arquivo ARQ1 antigo que estava no drive B permanece lá até que a cópia tenha sido totalmente efetuada sobre o arquivo temporário de trabalho. Somente após isso é que é apagado o arquivo ARQ1 antigo.

Se durante o processo de cópia houver uma queda de luz, ficará no disco algum arquivo com $$$ em sua extensão. Você poderá apaga-lo posteriormente (com o comando ERA) pois todos os arquivos que tenham esta extensão no nome são considerados arquivos de trabalho, não contendo portando informação confiáveis.

O programa PIP tem uma série de opções que podem afetar o arquivo durante o processo de cópia. Das várias opções existentes vamos mostrar as mais usadas.

Estas opções são informadas no final do comando e devem estar entre colchetes.

A opção V faz com que o PIP faça uma verificaçao da cópia. Quando a utilizamos, o programa PIP.COM faz uma verificação lendo aquilo que foi gravado contra o arquivo original. Caso haja alguma discrepancia (o que praticamente nunca acontece) o programa PIP emite uma mensagem de erro e abandona a cópia.

A cópia fica um pouco mais demorada, mas com certeza mais confiável.

```
A>PIP
*B:=A:PIP.COM[V]
*^C
A>
```

A opção O diz ao PIP que o arquivo não é do tipo Texto, fazendo com que o PIP copie todo o arquivo mesmo encontrando o caracter &H1A que indica seu fim. Quando o arquivo a ser copiado tem a extensão .COM, o PIP sabe que é um programa e que provavelmente encontrará &H1A no meio dele, ativando a opção O automaticamente. De qualquer forma, é sempre bom colocar estas duas opções, o V e o O, para garantir uma cópia bem feita.

```
A>PIP B:=A:*.*[OV]
```

Para evitar alguns dissabores com o tempo, é muito aconselhável utilizar estas opções sempre que utilizar o PIP.

A sequência em que são dadas as opções não importa. Podemos indicar OV como também VO que o efeito é o mesmo.

```
*B:=A:*.*[VO]
```

Alguns arquivos podem possuir características especiais, como vimos no programa STAT.COM . Estas características são os arquivos de sistema (SYS) e os arquivos protegidos contra alterações (R/O).

O programa PIP possui a opção R , que faz com que mesmo os arquivos de sistema sejam copiados. Se não for dada a opção R , serão copiados apenas os arquivos tipo DIR, e se for dada a opção R , serão copiados todos os arquivos.

Se voce utilizar esta opção sempre, poderá ficar sossegado quando for tirar uma cópia de todos os arquivos de um disco, pois com certeza serão copiados todos os arquivos mesmo, e não apenas aqueles que vemos com o comando DIR.

```
A>PIP B:=A:COPY.COM[OVR]
```

Uma outra opção muito utilizada é a opção W. Quando no disco que for gravado o arquivo destino já existir um arquivo com mesmo nome, o PIP apaga este arquivo para poder modificar o nome do arquivo temporário de trabalho.

Se este arquivo for R/O, o PIP envia uma mensagem perguntando se pode ou não apagar o arquivo R/O antigo para poder, então, finalizar o processo de cópia. Essa opção (W) diz ao PIP para não emitir essa mensagem, apagando diretamante o arquivo destino (se houver) mesmo sendo R/O.

```
A>PIP B:=A:PIP.COM[OVRW]
```

As opções até aqui descritas servem para a cópia tanto de arquivos como de programas. As demais opções que veremos agora deverão ser usadas apenas para arquivos de textos e nunca para programas. Se voce as utilizar em programas este serão copiados com algumas alterações e fatalmente deixarão de funcionar.

A opção F faz com que o PIP não copie os caracteres &H0C, que para o HB-MCP significa "avançar uma folha". Se

você possuir um arquivo de textos que contém este caractere, ao listar este arquivo (pelo TYPE, por exemplo), você notará que em alguns pontos haverá um salto de folha devido a existencia desse caractere. Se você quiser retirar esse caractere copie esse arquivo através do PIP e coloque a opção F. O arquivo copiado será igual ao arquivo original, porém sem saltos de folha (sem os caracteres &H0C).

A opção Pn faz com que o PIP insira o caractere &H0C após n linhas impressas.

Essa opção existe para podermos listar um arquivo na impressora sem nos preocupar com o picote do papel. Se, por exemplo, especificarmos P60 como opção, a cada 60 linhas impressas o PIP envia um caractere &H0C à impressora para esta avançar uma página.

Esta opção junto com a opção F forma uma boa dupla quando queremos imprimir um texto.

Qualquer caractere &H0C que já venha no arquivo será eliminado pela opção F e a cada n linhas impressas o próprio PIP se encarregará de enviar o caracter &H0C à impressora.

Geralmente utilizamos 60 na opção P, mas pode ser utilizados outros números como 50 por exemplo.

```
A>PIP LST:=A:TEXTO[FP60]
```

LST: representa a impressora no HB-MCP.

A opção N faz o PIP numerar as linhas. Quando usamos a opção N, o PIP envia primeiro o número da linha e depois a linha em sí, e esta numeração possui espaço para 6 dígitos, seguidos por ":" .

A opção N2 tem o mesmo efeito que a opção N, só que os ":" não aparecem.

Veja uma comparação do texto original presente no arquivo e o que será impresso com as opções N e N2 .

Texto original	Usando N	Usando N2
Agora um	1: Agora um	1 Agora um
computador	2: computador	2 computador
quebrou	3: quebrou	3 quebrou
na mesa	4: na mesa	4 na mesa
do diretor	5: do diretor	5 do diretor
e pode	6: e pode	6 e pode
ocasionar	7: ocasionar	7 ocasionar
um desastre	8: um desastre	8 um desastre
para o	9: para o	9 para o
departamento	10: departamento	10 departamento

A opção Tn faz com que o PIP substitua o caractere TAB (&H09) por tantos brancos quanto necessários para que o próximo caractere a ser copiado caia na coluna múltipla de n.

Esta opção existe porque certos periféricos não interpretam o caracter TAB.

Se o texto a ser copiado possuir este caractere, o

periférico que for receber a cópia não o interpretar, você ficará obrigado a utilizar esta opção. Ou então quando você quiser uma listagem em que as tabulações não sejam de oito em oito colunas como é o normal, poderá utilizar esta opção indicando de quanto em quanto se deseja a tabulação.

```
A>PIP LST:=B:TEXTO[FP60T10N2]
```

A opção Z faz com que o bit 7 de cada byte copiado seja zerado.

Todos os caracteres da tabela ASCII contém o sétimo bit de cada byte zerado, permitindo assim apenas 128 caracteres. Alguns editores de texto (como o Wordstar por exemplo) aproveitam então esse sétimo bit como bit de controle, fazendo com que apareçam caracteres fora da tabela ASCII (caracteres acentuados, figuras etc.). Fazendo uma cópia desse arquivo com a opção Z, o arquivo volta a ter apenas caracteres ASCII.

A opção U faz o PIP transformar todas as letras minúsculas em letras maiúsculas durante a cópia, e a opção L faz justamente o contrário.

Com estas duas opções pode-se imprimir um texto apenas com letras maiúsculas mesmo que este texto tenha sido digitado com letras minúsculas e vice-versa.

O PIP pode copiar apenas parte do arquivo original, podendo ser especificado o início e o fim do texto a ser copiado.

A opção S indica o texto a partir de onde o PIP começará a cópia e a opção Q indica o texto final até onde o PIP fará a cópia. Após o texto indicador de início ou fim do bloco, deve-se teclar ^Z.

```
*LST:=A:TEXTO[SASSIM^ZQENTAO^Z]
```

Com este comando será listado todo o texto compreendido entre a primeira ocorrência da palavra ASSIM até a primeira ocorrência da palavra ENTAO após o início da impressão. Caso não seja encontrados este delimitadores (ASSIM e ENTAO neste exemplo) o programa PIP envia uma mensagem de erro.

Podemos também ver o que o PIP está copiando colocando a opção E.

```
A>PIP A:=B:TEXTO[E]
```

Finalmente podemos também juntar vários arquivos através do PIP. Basta separá-los por vírgula.

```
A>PIP A:TODOS=B:ARQ1,B:ARQ2[VO]
```

Como você pode ver, o programa PIP.COM tem muitos recursos para se copiar arquivos e programas de um disco para outro desde que se tenha dois drives.

Além das opções aqui apresentadas o PIP contém ainda outras opções que servem apenas quando programamos em Assembly, surgindo assim alguns arquivos com formato especial e que o PIP os reconhecem, mas que são o tanto complicados para mostrarmos aqui, mesmo porque todas as opções apresentadas são mais do que suficientes para copiarmos bem os nossos arquivos.

Se você possui dois drives, é uma boa hora para tirar pelo menos uma cópia dos programas que vieram com o disco HB-MCP para um outro disco apenas como precaução.

Outra boa sugestão é a de ter os programas PIP.COM e STAT.COM em todos os os disquetes que forem usados com o sistema operacional HB-MCP.

Voce dominará os programas PIP e STAT somente com algum uso e experiencias.

Com um disco contendo uma cópia destes programas use e abuse deles, vendo como atua cada opção em separado, junto com outras etc. somente assim você realmente dominará o micro : utilizando !!!!

COPY.COM

Se você tem apenas um drive, já sabe que com o programa PIP não é possível copiar arquivos de um disco para outro. O programa COPY.COM foi elaborado justamente por causa disso. Ele consegue copiar arquivos de um disco para outro mesmo tendo apenas um drive. Para executá-lo basta digitar o seguinte comando:

```
A>COPY nome do arquivo
```

O nome do arquivo é opcional e, se não for especificado, o vídeo ficará assim:

FIGURA 3.30 - Dados fornecidos pelo comando COPY.

```
COPY
Copia arquivos
V1.0

Copia todos (S/N) ?
```

Se a resposta for "N" , o COPY será cancelado e o A> voltará ao vídeo. Se você responder "S" , então todos os arquivos serão copiados (figura 3.31).

Nesse ponto, coloque o disco a ser copiado e tecle RETURN. O programa COPY irá ler tantos arquivos quantos couberem na memória do micro. Quando ela estiver repleta, ele pedirá que se introduza no drive o disco que receberá a cópia para poder então gravar nele os arquivos lidos (figura 3.32).

Quando o nome do arquivo é especificado, o COPY pede que seja inserido o disquete origem e que se pressione a te-RETURN. A partir daí a cópia do arquivo será executada.

FIGURA 3.31 - Copiando todos os arquivos com o COPY.

```
COPY
Copia arquivos
V1.0

Copia todos (S/N) ? S

Coloque disquete origem e
Tecle <RETURN>
```

FIGURA 3.32 - Solicitação de troca de disco pelo COPY.

```
Lendo arquivo:     AUTOLD   .COM
Lendo arquivo:     BACKUP   .COM
Lendo arquivo:     TERM     .COM
Lendo arquivo:     DSKCNV   .COM
Lendo arquivo:     BATCH    .COM
Lendo arquivo:     DROUT    .COM

Coloque disquete destino e
Tecle <RETURN>
```

Se durante o processo de cópia de algum arquivo, este já existir no disco destino, então o programa COPY perguntará se deve eliminar ou não o arquivo antigo.

FIGURA 3.33 - Arquivos existentes no disco de destino.

```
Coloque disquete destino e
Tecle <RETURN>
Gravando arquivo: PIP     .COM
Arquivo existe. Elimina (S/N) ?
```

Se após ter gravado todos os dados lidos ainda restarem mais arquivos a copiar, o programa COPY pedirá que seja inserido novamente o disco de origem para continuar o processo de cópia. Assim ocorre até que tenham sido copiados todos os arquivos. Se a sua resposta a primeira pergunta feita pelo COPY for "N", serão copiados apenas os arquivos que você especificar.

Caso você possua dois drives, verá que é mais fácil utilizar o programa PIP.COM porque não necessita trocar disquetes, mas com apenas um drive o COPY.COM terá que ser utilizado e nada melhor do que o uso constante para dominar totalmente sua maneira de utilização. Apenas tome muito cuidado com o troca-troca de disco para não danificar o disco a ser copiado !!!

Se o drive E: estiver configurado, o COPY apresentará algumas opções para que você escolha o drive origem e o drive destino, possibilitando o uso do COPY com dois drives!

DSKCNV.COM

Já sabemos que cada fabricante usa uma formatação de discos que lhe é mais conveniente, acarretando uma certa incompatibilidade de disquetes entre seus micros. Atento a isso, o seu HOTBIT com o HB-MCP vem com um utilitário chamado DSKCNV.COM que prepara o drive E: de acordo com a formatação que você escolher. O uso do drive E: é apenas lógico, pois o drive realmente usado é o A: (quando você tem apenas um drive) ou o B: (quando você tem dois drives).

Assim, se você já possuir disquetes gravados em outros micros (ITAUTEC I-7000, PROLOGICA S-700) poderá utilizá-los sem preocupações. Pa;a tanto, use o seguinte comando:

```
A>DSKCNV
```

E então será mostrada a seguinte tela:

```
DSKCNV
Converte Drive Lógico
V1.0

(A) 5.25 FS HB8000
(B) 5.25 FD HB8000
(C) 3.50 FS HB8000
(D) 3.50 FD HB8000
(E) 5.25 FS S700
(F) 5.25 FD S700
(G) 5.25 FS I7000
(H) 5.25 FD I7000
(I) 5.25 FD MZ3500
(J) 5.25 FD SVI707

(Z) Não Configura

Escolha Opção:
```

Escolha então a opção adequada conforme o menu apresentado. Feito isso, ao acessar o drive E:, você verá que o drive A: ou B: é que realmente será acessado, mas internamente, os dados serão lidos na formatação indicada.

No "menu" apresentado, os números 5.25 e 3.5 representam, respectivamente, drives para os discos de 5 1/4" e 3 1/2"; FS significa face simples e FD face dupla.

Os nomes que aparecem logo após referem-se ao tipo de equipamento : HB8000 para o HOTBIT, CP700 para PROLOGICA, I7000 para ITAUTEC, etc.

É conveniente copiar os programas e arquivos que você possuir em discos gravados por outros micros para o seu

HOTBIT, copiando os arquivos do drive E: para o drive A: com o programa PIP.COM ou COPY.COM (se você tem apenas um drive), e evitando assim aquele "carnaval" de formatações diferentes em seus discos. Note que, uma vêz configurado pelo DSKCNV, o drive E: pode ser usado pelo COPY.COM .

AUTOLD.COM

Com o tempo você verá que é muito prático separar os disquetes em grupos de acordo com os programas que você tem. Um grupo de disquetes só para jogos, outro só para processamento de textos e assim por diante, mesmo que isso cause um aparente desperdício de disco. Fazendo essa separação, quando você for utilizar o micro com uma certa finalidade irá pegar os discos correspondentes de forma organizada, sem ter que ficar procurando um programa ou arquivo em todos os discos que você possuir!!!

Os produtores de software já preparam os discos dessa maneira, de modo que ao utilizar um sistema de contabilidade, por exemplo, você use sempre o mesmo disco e digite sempre o mesmo comando.

Sendo assim, seria muito cômodo se o computador "adivinhasse" qual o disco que estamos inserindo no drive e já desse o comando adequado automaticamente. O HOTBIT faz quase isso.

O programa AUTOLD.COM permite que seja executado um comando assim que é carregado o HB-MCP, antes mesmo de ser visualizado o "A>".

Para usá-lo, digite o comando desejado logo após o seu nome, por exemplo:

```
A>AUTOLD DIR
```

Você obterá a seguinte tela:

```
AUTOLD
Auto-execução de comando
V1.0


Comando:  DIR




Pode gravar no disco (S/N) ?
```

Respondendo "s", toda vez que se der um RESET com este disco no drive A e se entrar no HB-MCP, será executado um comando DIR antes de aparecer o "A>".

FIGURA 3.34 - DIR produzido pelo AUTOLD.

```
HB-MCP  [ EPCOM ]  V1.0

A: PIP      COM : FORMAT   COM
A: DRINP    COM : STAT     COM
A: XSUB     COM : SUBMIT   COM
A: FUNCOES  COM : COPY     COM
A: DUMP     COM : COPDOS   COM
A: SYSGEN   COM : AUTOLD   COM
A: BACKUP   COM : TERM     COM
A: DSKCNV   COM : BATCH    COM
A: DROUT    COM
A>
```

Se você responder "n" à pergunta feita pelo programa AUTOLD.COM ele nada faz, sendo mostrado então o "A>".

FIGURA 3.35 - Opção "n" com o AUTOLD.

```
AUTOLD
Auto-execução de comando
V1.0

Comando:  DIR



Pode gravar no disco (S/N) ? n

AUTOLD cancelado
A>
```

Se você quiser desfazer essa execução automática de um certo comando, basta executar o programa AUTOLD.COM sem nenhum parâmetro e responder "S" à pergunta:

```
A>AUTOLD
```

Será apresentada a seguinte tela:

FIGURA 3.36 - Desativando a execução automática do AUTOLD .

```
AUTOLD
Auto-execução de comando
V1.0

Comando:



Pode gravar no disco (S/N) ? s
A>
```

Pode-se usar qualquer comando com o programa AUTOLD, como por exemplo: DIR, SUBMIT, nome de programa, etc.

Com esse recurso, toda vez que for colocado um certo disco no drive A e for dado RESET no micro, pode-se fazer com que um certo programa já entre em execução automaticamente, facilitando muito o trabalho com o computador.

DUMP.COM

O programa DUMP.COM serve para visualizarmos no formato hexadecimal todos os dados gravados em um arquivo e, se possível, seus caracteres correspondentes (em MSX-BR).

Com esse programa, é possível vermos todos os dados gravados num arquivo, incluindo os caracteres acentuados, figuras, etc, só que no formato hexadecimal, de uma forma mais cômoda que aquela apresentada pelo comando TYPE.

O nome do arquivo a ser listado deve ser fornecido logo após o comando DUMP:

```
A>DUMP DUMP.COM
```

Sendo então mostrada a seguinte tela:

FIGURA 3.37 - Tela do comando DUMP.

```
DUMP V1.0  [Arquivo = DUMP    .COM]

0000:18 0D 5B 56 31 2E 30 2F ..[V1.0/
0008:30 30 36 45 44 55 5D ED 006EDU]
0010:73 F8 02 31 4B 03 0E 00 sò.1K...
0018:11 6D 00 1A FE 24 20 12 .m.. $ .
0020:13 1A FE 49 0E 01 28 06 .. I..(.
0028:FE 44 20 06 0E 02 79 32  D ...y2
0030:F5 02 3A F5 02 CB 47 20  .: . G
0038:0B 3A 03 00 E6 03 FE 01 .:..€.  .
0040:3E 07 28 02 3E 0F 32 FA >.(.>.2ú
0048:02 11 B5 02 21 5C 00 7E ...!\.
0050:B7 23 28 08 C6 40 12 13 u#(. @..
0058:3E 3A 12 13 01 08 00 ED >:.....
0060:B0 3E 2E 12 13 01 03 00 â>......
0068:ED B0 3E 5D 12 CD 49 02 â>]. I.
0070:CD 82 02 FE FF 20 0E 11  .   ..
0078:C9 02 CD 55 02 CD 15 02  . U. ..
0080:ED 7B F8 02 C9 AF 32 F7 {ò.  2=
0088:02 32 F6 02 21 00 00 3A .2/.!..:
0090:F6 02 3C 32 F6 02 FE 15 /.<2/. .
0098:38 19 3E 01 32 F6 02 3A 8.>.2/.:

Aperte uma tecla
```

Se você quiser cancelar a execução do programa, basta teclar ^C em resposta à frase "Aperte uma tecla".

Os números que aparecem à esquerda dos dois pontos (:) servem apenas para nos basearmos onde os dados estão posicionados no arquivo em relação ao seu início.

Os pares de números que aparecem após os dois pontos

representam, em hexadecimal, o valor do byte que está gravado no arquivo e, mais a direita, está a representação em ASCII (se possível) desses dados.

Os códigos de controle (de 0 a &H1F) são mostrados como um ponto (.) .

Com esse utilitário, fica fácil imaginar o que um certo programa faz simplesmente "vendo" as mensagens que ele possui internamente. Também é de extrema utilidade quando estamos testando algum programa qualquer que grava dados em arquivo para analisarmos os dados que este programa gravou.

DROUT e DRINP

No padrão ASCII existem 128 caracteres, cada um com um código numérico entre 0 e &H7F. Como é possível gerar 256 códigos diferentes com apenas 1 byte, restam outros 128 códigos que não estão na tabela ASCII. O HOTBIT usa esses códigos para caracteres acentuados, cedilhas, figuras, etc.

Aqui no Brasil existem ao menos dois outros padrões, o ABNT e o ABICOMP, que englobam inclusive alguns caracteres de códigos maiores que &H7F mas que não equivalem aos códigos do padrão MSX. Os programas DROUT.COM e DRINP.COM servem como "compatibilizadores" entre os caracteres que entram no micro e os que saem do micro, de acordo com o padrão escolhido.

A letra "á" no seu HOTBIT tem código &HA0. Essa mesma letra tem o código &HC2 no padrão ABICOMP.

Com o utilitário DROUT.COM, é possível teclarmos "á" no HOTBIT e ser impresso "á" em uma impressora que aceite o ABICOMP. O mesmo vale para o programa DRINP.COM quando seu HOTBIT estiver acoplado a uma interface RS-232C.

Para reconfigurarmos os caracteres que saem do HOTBIT usamos o seguinte comando:

```
A>DROUT
```

Com isso obtemos a tela da figura 3.8 .

FIGURA 3.38 - Tela gerada por DROUT .

```
DROUT
Converte caracteres na saída
V1.0

(A) Espaços para códigos
    acima de 7Fh

(B) Padrão ABNT ("default"
    do HB-MCP)
    Padrão HOT-BASIC V1.2

(C) Padrão HOT-BASIC V1.1

(D) Padrão MSX ou Relação
    direta de códigos

(Z) Não altera
```

Escolhe-se então a opção mais adequada.

Para reconfigurarmos os caracteres para a entrada, usamos o programa DRINP.COM com o seguinte comando:

```
A>DRINP
```

Aparece, então, a seguinte tela:

FIGURA 3.39 - Tela do DRINP .

```
DRINP
Converte caracteres na entrada
V1.0

(A) Espaços para códigos
    acima de 7Fh

(B) Padrão ABNT ("default"
    do HB-MCP)
    Padrão HOT-BASIC V1.2

(C) Padrão HOT-BASIC V1.1

(D) Padrão MSX ou Relação
    direta de códigos

(Z) Não altera

 Opção:
```

Escolhe-se então a opção mais apropriada.

Após ter executado estes dois programas você poderá, por exemplo, listar qualquer programa ou arquivo que contenha letras acentuadas que elas serão impressas normalmente caso a sua impressora pertença à um dos padrões mostrados nos exemplos anteriores.

FUNCOES

Você tem no HOTBIT 10 teclas de funções marcadas de F1 a F10.

Nessas teclas podemos armazenar os comandos mais utilizados, simplificando muito o trabalho de digitação. Se uma certa sequência de caracteres é muito utilizada, pode-se programar a tecla F1, por exemplo, para que toda vez que ela for pressionada, produza um efeito equivalente ao que obteríamos se digitássemos a sequência de caracteres.

Quando o HOTBIT é ligado e o HB-MCP é carregado, as teclas F1 a F10 já vem pré-programadas com os comandos mais usados. Mas nem sempre estes comandos são realmente os mais usados por você!!

O programa FUNCOES mostra como reprogramar as teclas de F1 a F10, e também a atual configuração dessas teclas.

Digite então :

A>FUNCOES

E deverá aparecer a seguinte tela :

FIGURA 3.40 - Tela do comando FUNCOES .

```
FUNCOES
Teclas de Funções
V1.0

<CODE> + <função>  Programa Função
<ESC> Encerra / Máximo 15 Caracteres

F1= FUNÇÕES ^M
F2= DIR ^M
F3= STAT *.* ^M
F4= ERA
F5= PIP

F6= TERM ^M
F7= DIR B: ^M
F8= STAT B:*.* ^M
F9= DRINP ^M
F0= DROUT ^M

A>
```

Para reprogramar as teclas F1 a F10, deve-se proceder da seguinte maneira: Mantendo a tecla <CODE> apertada, pressiona-se a tecla de função a ser reprogramada. Se for pressionada <CODE> + F1 para se reprogramar a tecla F1 por exemplo, você verá o seguinte:

A>1=

Esta figura quer dizer que será reprogramada a tecla F1. Digite então todo o texto desejado, teclando por último a tecla (ESC). Por exemplo:

FIGURA 3.41 - Exemplo de reprogramação com o comando FUNCOES.

```
DIR    <RETURN>

A>1=DIR <RETURN> <ESC>
A>
```

Após ter sido digitado tudo isso, comando DIR será executado a simples digitação da tecla F1, pois foi associado à ela o comando DIR e o caractere &H0D (^M = RETURN).

O texto para cada tecla de função pode ter no máximo 15 caracteres.

Se uma tecla de função já tem o <RETURN> (como no exemplo anterior), ele será mostrado como ^M pelo utilitário FUNÇOES.

Esta nova configuração só permanece ativa enquanto o HOTBIT continuar ligado. Se for dado RESET ou o micro for desligado, as teclas de função voltarão a receber os comandos originais.

SUBMIT.COM

O utilitário SUBMIT.COM permite que sejam executados uma série de comandos do HB-MCP a partir de um arquivo, ao invés do teclado.

Se tivermos uma série de comandos que digitamos toda vez que formos rodar um programa, é conveniente gravarmos estes comandos em um arquivo para depois serem processados.

Vamos supor um programa chamado PROG1.COM que necessite dos seguintes comandos para serem executados:

FIGURA 3.42 - Exemplo de comandos de uso conjunto.

```
A>PIP B:ARQ1.BAK=A:ARQ1[OV]
A>PROG1
A>ERA ARQ1
A>REN ARQ1=ARQ2
A>
```

Seria muito trabalho digitarmos todos esses comandos um de cada vez sempre que os necessitássemos. Podemos então criar um arquivo com todos eles e sempre que quisermos rodar o programa PROG1.COM faremos com que o HB-MCP passe a aceitar tais comandos deste arquivo ao invéz do teclado. Vamos chamar este arquivo de PROCESS.SUB. O nome desse arquivo pode mudar mas a extensão .SUB é obrigatória, indicando que ele contém comandos reconhecidos pelo HB-MCP. Ele ficaria assim:

FIGURA 3.43 - Arquivo para uso do HB-MCP.

```
PIP B:ARQ1.BAK=A:ARQ1(OV)
PROG1
ERA A:ARQ1
REN A:ARQ1=A:ARQ2
```

Podemos criar este arquivo com o próprio PIP.COM :

```
A>PIP A:PROCESS.SUB=CON:
PIP B:ARQ1.BAK=A:ARQ1(OV)
PROG1
ERA A:ARQ1
REN A:ARQ1=A:ARQ2
^Z
*^C
A>
```

Quando quisermos executar o programa PROG1.COM, bastará dar o seguinte comando:

A>SUBMIT PROCESS

O Programa SUBMIT.COM cria um arquivo especial chamado $$$.SUB que contém os mesmos comandos do PROCESS.SUB, mas de uma maneira toda especial que faz o HB-MCP aceitar os comandos desse arquivo em vez de recebê-los do teclado.

Esse tipo de processamento é conhecido como processamento em lote, ou tipo BATCH.

Ao utilizar o programa SUBMIT.COM, você irá reparar que o HB-MCP acessará o disco, receberá um comando e passará a executá-lo. Este comando pode ser qualquer um que você daria normalmente pelo teclado.

Após terem sidos executados todos os comandos, o arquivo original PROCESS.SUB continua gravado no disco, e sempre que você precisar poderá chamar o programa SUBMIT.COM novamente.

Se você quiser interromper essa execução automática de comandos, basta pressionar qualquer tecla.

Você pode usar um processamento em lote todas as vezes que precisar dar sempre os mesmos comandos quando for executar algum programa. Basta criar um arquivo contendo os comandos na mesma ordem e com a mesma sintaxe que seriam usados pelo teclado e executar o programa SUBMIT.COM dando como parâmetro o nome do arquivo que contém tais comandos.

XSUB.COM

Após algumas experiências com o programa SUBMIT.COM você irá reparar que só quando digitarmos um comando para o CCP o HB-MCP acessa o disco para ler o próximo comando BACTH.

Na verdade, apenas o CCP passou a aceitar comandos de um arquivo ao invés de recebê-los pelo teclado.

Se durante a execução de um programa qualquer este precisar receber dados, eles terão que ser introduzidos pelo teclado.

Com o auxílio do programa XSUB.COM isso já não acontece. Esse programa é uma extensão do SUBMIT.COM e portanto só pode ser utilizado após o SUBMIT ter sido executado.

Vamos supor que o programa PROG1.COM peça uma data pelo teclado no formato dd/mm/aa. Com o programa SUBMIT.COM apenas os comandos foram fornecidos pelo arquivo PROCESS.SUB mas a data pedida pelo programa tinha que ser introduzida pelo teclado.

Com o auxílio do programa XSUB.COM isso já não acontece. Esse programa ao ser rodado dentro do SUBMIT.COM faz com que certas entrada de dados que venha do teclado passe a ser dada pelo arquivo usado pelo SUBMIT.

No exemplo anterior, podemos modificar o arquivo PROCESS.SUB para ficar assim:

FIGURA 3.44 - Arquivo para o XSUB.COM.

```
PIP B:ARQ1.BAK=ARQ1[OV]
XSUB
PROG 1
01/10/86
ERA A:ARQ1
REN A:ARQ1=A:ARQ2
```

Quando executado, o programa XSUB fará com que ao ser pedida a data pelo programa PROG1.COM, esta seja fornecida pelo próprio PROCESS.SUB (no exemplo, 01/10/86), já não sendo mais necessário digitá-la pelo teclado.

CARACTERES DE CONTROLE

No HB-MCP existem alguns caracteres de controle que são obtidos através do teclado mantendo a tecla "CONTROL" apertada enquanto se pressiona uma outra tecla.

O símbolo "^" antes de uma letra significa CONTROL. Por exemplo, ^X significa: pressionar a tecla CONTROL e, mantendo-a pressionada, digitar a tecla X.

Alguns desses controles facilitam a correção de uma linha quando digitamos algo errado nela.

Note porém que os controles que vamos ver agora nem sempre dão os resultados que esperamos !

Estes controles atuam apenas em uma rotina interna do HB-MCP, e essa rotina é a responsável pelo recebimento de uma linha pelo teclado.

Alguns programas quando necessitam receber algum comando (ou algum parâmetro) requisitam essa rotina interna do HB-MCP e recebem a linha já pronta, sem ter nenhum trabalho. Nestes casos os comandos que veremos atuam como esperado. Mas alguns outros programas possuem internamente sua própria rotina de digitação de uma linha e pode ser que os comandos que ele programa aceita não coincidam com os comandos aceitos pelo HB-MCP.

Os comandos que veremos agora podem ser dados quando o CCP aguarda ser digitado um comando, ou seja, quando vemos o "A>".

^C - TROCA DE DISCOS

Toda vez que for trocado um disquete, deve-se teclar ^C para informar ao CCP sobre a troca, portanto ^C só deve ser usado quando se tiver o "A>" (ou "B>",etc) e quando o disquete for trocado, senão na primeira vez que o disco for acessado e o HB-MCP perceber que foi trocado um disquete sem ter

sido usado o ^C, ele emitirá uma mensagem de erro, não permitindo o acesso ao disco para gravação.

Quando o HB-MCP acessa o disco pela primeira vez, ele pesquisa o diretório para obter uma série de informações sobre o quanto do espaço disponível do disco já foi utilizado e monta algumas tabelas internas na memória.

Toda vez que for necessário gravar algum dado no disco, o HB-MCP pesquisa essas tabelas para saber onde pode ser gravada a informação sem perder nada do que já esta gravado no disco.

Quando tiramos um disco e colocamos outro em seu lugar, o HB-MCP invalida essas tabelas, pois elas pertenciam ao disco que foi retirado e não ao disco presente.

Com essas tabelas invalidadas não é mais possível gravar informações no disco. Pode-se apenas lê-las.

Quando teclamos ^C, essas tabelas são refeitas e o CCP entra em ação (caso já não esteja).

Portanto, sempre que você trocar de disco e estiver visualizando o "A>", tecle ^C, senão nada poderá ser gravado nesse disco.

Alguns programas que pedem que sejam trocados os discos (COPY.COM por exemplo) já dão eles mesmos o ^C, não sendo necessário ser dado (e nem aconselhável!) um outro ^C dado por nós mesmos.

Sob outras circustâncias, o ^C é utilizado também para se cancelar o programa em execução, portanto tome cuidado ao usar ^C.

^E - CONTINUAR A DIGITAÇÃO NA PRÓXIMA LINHA

Para continuar a digitar um comando muito extenso na linha seguinte, pode-se teclar ^E.

O cursor passará para a linha abaixo e você poderá continuar a digitar o comando.

Quando finalmente for dado o RETURN, então toda a linha de comando será aceita, embora aparentem ser várias linhas, elas são na verdade apenas uma.

^H - RECUAR O CURSOR UMA POSIÇÃO

Quando errar na digitação de uma tecla, utilize ^H para recuar o cursor e digitar a tecla correta.

Você pode usar a tecla BS ◄◄ para obter o mesmo efeito.

^M - TÉRMINO DE UMA LINHA

^M tem o mesmo efeito que a tecla RETURN.

^P - LIGAR / DESLIGAR O "ECO" NA IMPRESSORA

Os caracteres que são enviados ao vídeo podem ser também enviados para a impressora.

Com esse recurso podemos tirar algumas cópias impressas das mensagens apresentadas por alguns programas.

Teclando ^P ligamos o "eco".

Teclando ^P novamente desligamos o "eco".

Se a impressora não estiver ligada ou estiver mal conectada ao micro, pode ocorrer do sistema ficar travado. Tome cuidado portanto ao utilizar o ^P.

^R - REPETIR UMA LINHA DIGITADA

Você pode dar ^R para ver melhor a linha que realmente será aceita ao ser dado o RETURN. Este comando apenas reapresenta a linha sendo digitada.

^S- PARA "TRAVAR" O VÍDEO

^S evita o rolamento, muitas vezes rápido demais, do vídeo. Sempre que se teclar ^S o vídeo ficará "travado" até que seja digitado outro ^S ou qualquer outra tecla.

^U ou ^X- LIMPAR A LINHA QUE ESTA SENDO DIGITADA

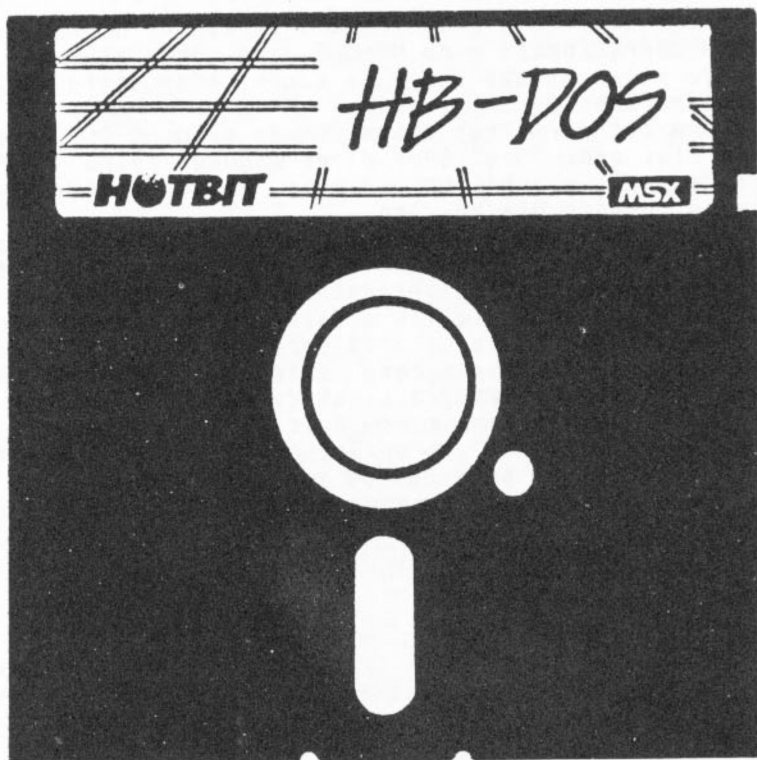
Quando você quiser desistir da linha que está digitando, tecle ^X que esta será totalmente apagada.

TECLAS ESPECIAIS NO HB-MCP

No HB-MCP, algumas teclas do HOTBIT receberam funções diferentes daquelas que você já conhece no DSK-BASIC. Veja na tabela a seguir as teclas especiais usadas no HB-MCP e respectivos códigos de controles que elas passaram a fornecer.

TECLA	COMANDO	TECLA	COMANDO
STOP	^S	SELECT	ESC
F1 a F10	Reconfiguráveis	CONTROL + STOP	RESET
HOME	^K + (RETURN)	^-	º
CLS	^L + (RETURN)	^=	ª
INS	^V	SHIFT +INS	^F
DEL	^G	SHIFT + DEL	DEL
▲	^E	SHIFT + ▲	^R
▼	^X	SHIFT + ▼	^C
▶	^D	SHIFT + ▶	^F
◀	^S	SHIFT + ◀	^A
CONTROL + ▲	ESC + A	CONTROL + ▶	ESC + C
CONTROL + ▼	ESC + B	CONTROL + ◀	ESC + D
^9	abre chave	^´ (agudo)	crase
^0	fecha chave	^¨ (trema)	trema
^~	til		

CAPÍTULO IV



O QUE É HB-DOS

HB-DOS é um sistema operacional desenvolvido especialmente para os micros da linha MSX.

Devido à grande popularidade do CP/M, o HB-DOS foi feito para ser, até certo ponto, compatível com ele,garantindo assim um grande número de programas já prontos para serem usados. Além disso, é também um pouco semelhante ao MS-DOS, um sistema operaciomal bastante poderoso desenvolvido pela MICROSOFT (R) para o IBM-PC (R).

Devido à semelhança entre o HB-MCP e o HB-DOS, vamos traçar um paralelo entre os dois no decorrer deste capítulo, mostrando algumas vantagens e desvantagens de cada um deles.

Ao se ligar o micro com um disco contendo o HB-DOS, é apresentada a seguinte mensagem:

```
HB-DOS versão V1.0
    [ EPCOM ]

HB-CCP versão V1.0

Data corrente: Qua 1/01/1986
Entre c/ nova data:
```

Logo de início é mostrada a versão do HB-DOS (que é o sistema operacional) e do HB-CCP, que seria equivalente ao CCP, aquele programa que serve de comunicador entre você e o sistema operacional.

Uma das primeiras ações tomada pelo HB-DOS é mostrar a data do dia, onde "Seg" quer dizer segunda-feira,"Ter" quer dizer terça-feira e assim por diante, sendo seguida da data completa no formato Dia/Mês/Ano.

Essa data não é a correta, pois o HOTBIT não possui um relógio interno e, sabendo disso, ele assume 01/01/86 como sendo o dia corrente e lhe pergunta qual a data correta. Nesse instante, você pode digitar a data que desejar ou simplesmente teclar RETURN (e nesse caso será assumido 01/01/86).

Para se digitar a nova data, basta digitar então o dia (com dois algarismos), o sinal de menos "-" ou uma barra de divisão "/"; o mês,também com dois algarismos, outra barra "/" ou sinal de menos "-"; e então, o ano com dois algarismos ou quatro algarismos, seguidos da tecla RETURN.

Veja um exemplo, supondo que hoje é 28/06/86:

```
HB-DOS versão V1.0
    [ EPCOM ]

HB-CCP versão V1.0

Data corrente: Qua 1/01/1986
Entre c/ nova data: 28/06/86

A>
```

Se você errar na digitação da data utilize a tecla BS para fazer as correções. Se for digitada uma data inválida, você será novamente solicitado a fornecer a data.

Observe que não foi necessário digitar o dia da semana (segunda, terça, etc) pois o HB-DOS descobre isso a partir da data digitada.

Logo após ter sido informada uma data correta, é apresentado o "A>", indicando que o HB-DOS está pronto para receber um comando seu. Repare que é o mesmo sinal mostrado pelo HB-MCP, a não ser pelo cursor, que não fica piscando no HB-DOS.

É recomendável sempre digitar uma data correta, pois sempre que um arquivo é alterado, o HB-DOS grava a data da alteração e esta é mostrada no comando DIR (que veremos mais adiante). Sendo assim, se for informada a data do dia corretamente sempre que o micro for ligado, ficará fácil saber quando foi que utilizamos um arquivo pela última vez, apenas utilizando o comando DIR.

Esta é uma das vantagens que o HB-DOS tem sobre o HB-MCP, pois fornece uma documentação a mais sobre o uso dos arquivos no disco.

DATE

Você pode alterar a data a qualquer hora que desejar com o comando DATE. Este comando serve para mudarmos a data que foi assumida quando o HB-DOS foi inicializado.

Veja um exemplo :

```
A>DATE                          <RETURN>
Data corrente: Sáb 28/06/1986
Entre c/ nova data:             <RETURN>

A>
```

Podemos usar o comando DATE para sabermos a data corrente sem precisar alterá-la, bastando simplesmente teclar RETURN quando for solicitada a nova data ou então fornecer efetivamente uma nova data.

Se é sua intenção modificar a data, você pode indicá-la logo após o comando DATE.

Veja como isso pode ser feito

```
A>DATE 22/07/86

A>
```

Observe que, neste exemplo, nada foi mostrado pelo HB-DOS, mas com certeza a data foi alterada.

DIR

Quando o "A>" é mostrado, o HB-DOS está aguardando um comando seu, semelhantemente ao HB-MCP.

O comando DIR do HB-DOS é mais completo que o DIR do HB-MCP (visto no capítulo anterior), mostrando o nome do arquivo, seu tamanho (em bytes) e a data do último acesso. Após mostrar todos os arquivos, apresenta o espaço disponível no disco.

O comando DIR do HB-DOS é como se fosse uma mistura do comando DIR do HB-MCP e do programa STAT.COM

Duas opções foram implantadas ainda no comando DIR: /P e /W.

Se for colocado /P após o comando DIR, os arquivos serão mostrados de 23 em 23 linhas, e será solicitado o pressionamento de qualquer tecla (exceto ^C).

Assim, se em um disco existirem muitos arquivos gravados, o comando DIR irá mostrá-los até preencher toda o vídeo e aguardará que você pressione alguma tecla para poder continuar a mostrar os outros arquivos até preencher outra tela, e assim por diante. Portanto, não é necessário o uso do ^S para se travar o vídeo enquanto vemos o que queremos.

A opção /W faz com que não seja mostrado nem o tamanho do arquivo e nem a data do último acesso, mostrando mais de um arquivo por linha, ficando igual ao DIR do HB-MCP.

Além do /P e /W, são também permitidos o ponto de interrogação (?) e o asterisco (*).

Veja, a seguir, alguns exemplos:

```
A>DIR
HBDOS     SYS     4480   7-21-86
HBCCP     COM     6400   7-04-86 16:27
        2 Arquivo(s)    347136 livres
A>
```

```
A>DIR/W
HBDOS     SYS     HBCCP     COM
        2 Arquivo(s)    347136 livres
A>
```

```
A>DIR *.COM/W
HBCCP     COM
        1 Arquivo(s)    347136 livres
```

TYPE

O comando TYPE é outro que mantém as mesmas características do HB-MCP. Ele mostra o conteúdo de um arquivo do disco no vídeo. Para usá-lo, basta digitar o nome do arquivo a ser pesquisado logo à sua frente.

Veja o exemplo a seguir:

```
A>TYPE ARQUI.BAS
```

Você pode cancelar a execução do TYPE teclando ^C ou ^STOP. Se você teclar ^S o vídeo ficará parado até que seja pressionada qualquer tecla.

REN ou RENAME

O comando REN (ou RENAME), apesar de ter a mesma função do comando REN do HB-MCP, tem uma sintaxe ligeiramente diferente dele.

No HB-DOS, o nome do arquivo a ser alterado vem antes, e depois o novo nome, de maneira inversa ao HB-MCP.

```
A>REN VELHONOM NOVONOM
A>RENAME ARQ1 NOVOARQ
```

Observe também que existe apenas um espaço em branco entre o nome antigo e o novo nome do arquivo, ao invés do sinal de igual (=) usado no HB-MCP. Apesar disso, o HB-DOS pode aceitar, além do espaço, a vírgula (,) , o ponto e vírgula (;) e o próprio sinal de igual.

MODE

O HB-DOS, por ser um sistema operacional feito sob medida para o HOTBIT, possui um comando que indica quantos caracteres poderão ser mostrados em uma linha do vídeo, de maneira análoga ao comando SCREEN e WIDTH do BASIC: o MODE. Ele existe devido ao fato de alguns televisores mostrarem uma imagem muito expandida, fazendo com que certos caracteres mais à esquerda ou mais à direita do vídeo acabem não sendo visualizados. Você deve indicar quantos caracteres deseja por linha logo após a palavra MODE, separando-os por um espaço em branco.

```
A>MODE 40

A>MODE 20
```

Observe que após a execução do comando, o vídeo é limpo, como acontece com o CLS do BASIC.

Se o cartão de 80 colunas estiver conectado ao mìcro quando o HB-DOS é carregado, o modo 80 colunas será automaticamente acionado. Uma vêz em outro modo, para voltar a usar o modo 80 colunas no HB-DOS bastará digitar:

```
A>MODE 80
```

ERASE ou DEL

Outro comando do HB-DOS semelhante ao do HB-MCP é o ERASE. Ele é idêntico ao comando ERA. Você pode, também, digitar DEL para obter o mesmo resultado.

Estes comandos servem para apagar um arquivo do dis-

co, podendo-se também usar opcionalmente o ponto de interrogação (?) e o asterísco (*). Por exemplo, imagine que é necessário eliminar de um disco todos os programa terminados por .BAS . O comando mostrado a seguir resolve esse problama:

```
A>ERASE *.BAS
```

BASIC

O comando BASIC coloca o DSK-BASIC em funcionamento. Dessa maneira, mesmo estando no HB-DOS, é possivel entrar no BASIC sem ter que desligar ou dar RESET no micro.

No comando BASIC, pode-se especificar, logo em seguida, o nome do programa que se deseja executar, não sendo necessário carregá-lo e esperar o surgimento do "Ok" na tela para executá-lo. Por exemplo, se você estiver no HB-DOS e quiser executar um programa de nome TESTMOV1.BAS, basta digitar o comando mostrado adiante:

```
A>BASIC TESTMOV1.BAS
```

Quando se fornece o nome de um programa logo após o comando BASIC no HB-DOS, este programa terá que ser necessariamente um programa BASIC. Se não for, será enviada uma mensagem de erro pelo DSK-BASIC.

```
HOT-BASIC versão  1.1
EPCOM [Z.F.MANAUS] 1985
Mem. Livre 24455
DSK-BASIC V1.0
Ok
```

Uma vez no DSK-BASIC, você poderá usá-lo normalmente, pois apesar de ter sido chamado do HB-DOS, nenhuma mudança perceptível ocorre.

Quando você desejar voltar para o HB-DOS, basta usar o comando CALL SYSTEM (ou _SYSTEM).

O comando CALL SYSTEM só funciona no DSK-BASIC se ele tiver sido chamado do HB-DOS. Quando isso ocorre, ele deixa indicado em algum lugar da memória que pode ser usado este comando. Se você ligar o micro sem nenhum disco no drive ou se o disco não contiver o HB-DOS, então automaticamente será selecionado o DSK-BASIC. Como na hora de ligar o micro não havia o HB-DOS no disco, não será possível entrar nele com o comando CALL SYSTEM. Mesmo que você coloque um disquete no drive contendo o HB-DOS, o DSK-BASIC não perceberá que ele existe e portanto não permitirá o uso do comando CALL SYSTEM.

Se você estiver operando no DSK-BASIC, desejando entrar no HB-DOS, e o comando CALL SYSTEM não estiver permitindo, a solução será desligar e ligar novamente o micro, ou então, dar RESET.

VERIFY

O comando VERIFY serve para ligar ou desligar um indicador de verificação. Todo acesso feito ao disco para gravação pode ser com ou sem uma verificação e o comando VERIFY é quem determina isso.

VERIFY ON faz com que seja feita uma verificação toda vez que for gravado algo no disco e VERIFY OFF faz com que não seja feita essa verificação. O acesso ao disco fica bem mais lento se for dado o comando VERIFY ON, mas fica também mais confiável.

Na versão 1.0 do HB-DOS o comando VERIFY não está implementado!

COPY

O comando COPY serve para fazermos cópias de arquivos. É como o programa PIP do HB-MCP só que mais modesto, e a ordem é inversa à do PIP, vindo primeiro o arquivo a ser cocopiado e depois o arquivo que receberá a cópia.

No HB-DOS,embora possuindo apenas um drive, é possível simular dois drives. Neste caso o drive A seria um disco e o drive B outro disco, semelhante ao comando COPY do DSK-BASIC.

O comando COPY do HB-DOS também simula esses dois drives, pedindo para que seja feita a troca dos discos quando necessário. Como o HB-DOS só carrega o sistema operacional, o comando COPY tem muita memória disponível para fazer a cópia, sendo muito mais rápido que o COPY do DSK-BASIC.

```
A>COPY A:ARQ1 B:ARQ2

Insira disco no drive B:
e pressione qualquer tecla
          1 Arquivo(s) copiado(s)
A>
```

Após ter sido feita uma cópia de um drive para outro, pode acontecer de você dar um comando e o HB-DOS pedir para que seja colocado o disco no drive A, mesmo já existindo um disco ali. É que após a cópia, o HB-DOS sabe que o disco presente no drive está operando como se estivesse no drive B. O comando que você der, se for assumido o drive A, fará com que o HB-DOS peça para ser substituído o disco do drive.

```
A>COPY A:ARQ1 B:ARQ2

Insira disco no drive B:
e pressione qualquer tecla
        1 Arquivo(s) copiado(s)
A>DIR

Insira disco no drive A:
e pressine qualquer tecla
```

O comando COPY serve também como "juntador" de arquivos. Você pode concatenar mais de um arquivo através desse comando, indicando com um sinal de mais (+) quais os arquivos que serão concatenados.

```
A>COPY ARQ1 + ARQ2 A:TODOS
```

No exemplo anterior, os arquivos ARQ1 e ARQ2 foram "juntados" em um só arquivo de nome TODOS. O comando COPY possui ainda três opções para a cópia. Estas opções são indicadas através da barra (/) e mais uma letra que é a opção desejada.

A opção /A indica que o arquivo em questão esta em ASCII, tendo portanto apenas caracteres ASCII, e assim que for encontrado o caractere &H1A, que determina fim de arquivo, deve ser terminada a cópia.

A opção /B indica que o arquivo contém dados que não não são apenas caracteres ASCII, e que, mesmo encontrando o caractere &H1A, deve continuar a cópia.

Para se ter uma margem de segurança maior, é aconselhável usar o comando VERIFY ON ou utilizar a opção /V sempre que for efetuar uma cópia de arquivos.

```
A>COPY ARQ1/A B:ARQCOP/V

A>COPY ARQ/B ARQ2/B/V

A>COPY ARQ1 + ARQ2 + ARQ3 B:JUNTOS
```

Você viu no capítulo 1 uma relação de alguns periféricos que o HOTBIT pode ter com o auxílio de cartuchos de expansão. O comando COPY aceita o periférico CON como conso-

le, podendo ser tanto o teclado quanto o vídeo.

Se você quiser ver o está gravado num arquivo sem usar o comando TYPE, pode usar o comando COPY.

```
A>COPY ARQ1 CON
```

Você pode também gravar os dados que digitar no teclado em um arquivo de disco.

```
A>COPY CON DIGTA
isto foi digitado
^Z
        1 Arquivo(s) copiado(s)
A>
```

Quando o COPY é utilizado como descrito no último exemplo, tudo o que for digitado será gravado no disco. Para terminar, digite ^Z (CONTROL+Z) que envia o caracter &H1A indicando fim do arquivo, consequentemente terminando a cópia.

O periférico PRN (ou LST) é a impressora e podemos listar um arquivo qualquer indicando um dos dois periféricos (LST ou PRN).

```
A>COPY ARQ1 PRN
```

No exemplo anterior, foi listado o conteúdo do arquivo ARQ1.

FORMATAÇÃO DO DISQUETE E CÓPIA DO SISTEMA OPERACIONAL

Tanto o HB-DOS quanto o DSK-BASIC utilizam a mesma formatação de disco. Isso significa que os arquivos gravados pelo DSK-BASIC podem ser acessados pelo HB-DOS e vice-versa, sem nenhum problema de compatibilidade.

Esse compartilhamento de arquivos só é possível porque eles utilizam o mesmo programa formatador, presente na própria interface de disco, não necessitando de um programa específico para formatação do disco, como no caso do HB-MCP.

Para se formatar, então, um disco no HB-DOS, basta dar o seguinte comando:

```
A>FORMAT
```

Após ser dado este comando, será executado o programa formatador de disco presente na interface, de maneira idêntica àquela que se processa quando usamos o comando CALL FORMAT no DSK-BASIC. Sendo assim, não vamos repetir toda aquela "lenga-lenga" que você já leu no segundo capítulo desse livro, pois tudo se passará como já visto, sendo necessário responder as mesmas perguntas feitas pelo micro, etc...

Após a formatação de um disco, é sempre bom copiar o sistema operacional para ele.

No caso do HB-DOS, o sistema operacional está gravado na forma de arquivos comuns e não em trilhas reservadas especificamente para isso. Sendo assim, para se copiar o sistema operacional HB-DOS de um disco para outro, basta fazer uma simples cópia de arquivos utilizando o próprio comando COPY do HB-DOS.

Você já deve ter reparado que só vieram dois arquivos com o disco HB-DOS: HBDOS.SYS e HBCCP.COM . Pois bem, são estes dois arquivos que necessitam ser copiados para o disco que acabou de ser formatado a fim de que se tenha uma cópia do sistema operacional HB-DOS nele.

Coloque então o disco HB-DOS no drive A e digite o seguinte comando :

```
A>COPY A:HB*.*/B B:
```

Se você possuir apenas um drive, após ler os dados dos arquivos e colocá-los na memória, o comando COPY pedirá que seja retirado o disco que está no drive A e que seja colocado outro disco, representando o drive B. Retire então o disco presente no drive e insira em seu lugar o disco que receberá a cópia. A seguir tecle RETURN. O comando COPY copiará nesse disco os dois arquivos HBDOS.SYS e HBCCP.COM, que nada mais são que o próprio sistema operacional HB-DOS.

EXECUCÃO AUTOMÁTICA DE UM PROGRAMA

No HB-DOS existe um nome de arquivo muito especial: AUTOEXEC.BAT .

Sempre que se entra no HB-DOS, após o micro ter sido ligado, ele verifica se existe algum arquivo com esse nome gravado no disco. Se esse arquivo existir, os comandos lá presentes serão automaticamente executados.

Por se tratar de um arquivo com a extensão .BAT , devem estar gravados os mesmos comandos que daríamos quando temos o A> na tela, pois se trata de um arquivo para processamento em lote.

Com isso, é possível que um certo programa seja executado sempre que um determinado disco com o HB-DOS for colocado no drive e o micro for ligado. As vantagens disso você já deve ter percebido ...

PROCESSAMENTO EM LOTE

Um poderoso recurso do HB-DOS é a possibilidade de processamento em lote. O processamento em lote (também conhecidos como "Batch") é a execução sequencial de comandos de um arquivo previamente preparado, em vez de comandos do teclado.

Primeiramente escrevemos todos os comandos em um arquivo. Esses comandos são os mesmos que podemos utilizar

quando visualizamos o "A>".

O arquivo que contém os comandos tem que ter apenas uma particularidade: a extensão .BAT para indicar que ele é um arquivo de comandos.

Após a preparação do arquivo, basta digitarmos seu nome para que o HB-DOS comece a executar os comandos que ele contém.

Vamos supor que temos um sistema de contabilidade no qual geralmente usamos um programa para a digitação (movimento) e um outro para a verificação dos dados indicando possíveis erros (consistência). Após a execução do programa de verificação, é executado um programa de atualização e depois outro que emite alguns relatórios. Fica um tanto incômodo termos que decorar a ordem e os nomes de cada um dos programas descritos, sem falar que é sempre bom tirarmos cópias de segurança de todos os arquivos utilizados. Em casos como esse o processamento em lote pode ser bastante útil.

Vamos supor que o programa de digitação se chamasse DIGIT, o de verificação VERIFIC, o de atualização ATUALI e o de impressão de relatórios de LISTAG.

Sempre que formos utilizar este nosso sistema, temos que digitar esses quatro nomes, o que acaba se tornando cansativo. Podemos então criar um arquivo contendo exatamente esses mesmos comandos para executá-los de uma só vez! Nosso arquivo ficaria estruturado da seguinte maneira:

```
COPY CADASTR COPCADS
COPY MOVTO COPMOVT
DIGIT
VERIFIC
ATUALI
LISTAG
```

Na primeira linha é utilizado o comando COPY para se tirar uma cópia do cadastro, e na segunda linha é feita uma cópia do movimento. Nas demais linhas, se sucedem os nomes dos programas do sistema de contabilidade.

Após a preparação desse arquivo (que vamos chamar de SIST.BAT) bastará digitarmos SIST para que o HB-DOS passe a executar os comandos lá gravados um a um, como se eles tivessem sido digitados, poupando-nos muito trabalho.

Para prepararmos o arquivo SIST.BAT podemos utilizar o própio comando COPY:

```
A>COPY CON SIST.BAT
COPY CADASTR COPCADS
COPY MOUTO COPMOUT
DIGIT
VERIFIC
ATUALI
LISTAG
^Z
```

Toda vez que digitamos algo quando aparece o "A>", o

HB-DOS verifica se o que foi digitado é algum comando interno. Se não for, ele procura no disco algum arquivo que tenha a extensão .COM (sendo portanto um programa) e se não encontrar, procura um arquivo com a extensão .BAT.

Sempre que digitarmos o nome de um arquivo que tenha a extensão .BAT, o HB-DOS passará a executar seus comandos.

Para cancelarmos um processamento em lote devemos teclar ^C. Feito isso, o HB-DOS envia uma mensagem perguntando se realmente queremos cancelar o processamento.

Existem dois comandos próprios para serem usados nos arquivos de lotes:

REM - semelhante ao comando REM do BASIC, este comando permite que sejam colocados comentários nos arquivos em lote.

PAUSE - produz uma pausa no processamento. Ao encontrar esse comando, o HB-DOS envia uma mensagem pedindo para se teclar qualquer tecla (exceto ^C) para prosseguir com o processamento em lote.

Podemos usar esses dois comandos quando um arquivo para processamento em lote contiver comandos que nem sempre devem ser executados. Com o comando REM, indica-se o que irá ser feito se o processamento continuar. Logo em seguida, pode-se usar um comando PAUSE para produzir uma pausa na execução a fim de que o operador decida se o processamento deve ou não continuar.

Outro ponto forte do processamento em lote no HB-DOS é a possibilidade de se utilizar parâmetros sinalizadores: é possível fazer um arquivo para processamento em lote contendo sinais especiais, sem nenhum significado para o HB-DOS, mas que poderão ser substituidos por comandos quando necessário.

Vamos tomar como exemplo o nosso sistema de contabilidade, mais precisamente aquele arquivo SIST.BAT .

Nas duas primeiras linhas existe o comando COPY, que nem sempre deverá ser executado. Para conseguimos que ele só seja executado quando o desejarmos, poderíamos ter outro arquivo de comandos (SIST2.BAT por exemplo) sem as duas primeiras linhas do arquivo SIST.BAT:

```
DIGIT
VERIFIC
ATUALI
LISTAG
```

Ao ser digitado SIST2, o HB-DOS passará a executar aqueles programas da mesma maneira que com o SIST, mas sem fazer as cópias dos arquivos. Ness caso, temos dois arquivos muito parecidos com a mesma finalidade: processar a contabilidade.

Através de parâmetros sinalizadores, podemos fazer apenas 1 arquivo que atenda a duas necessidades. Nesse arquivo, no lugar do comando COPY nas duas primeiras linhas, colocamos um sinal dizendo ao HB-DOS que ali será inserido um co-

mando que poderá variar a cada vez que for executado o processamento em lote. Este sinal é o símbolo de porcentagem (%). Como o HB-DOS aceita até dez parâmetros de sinalização, é preciso colocar o número do parâmetro logo após o símbolo de porcentagem (%0 a %9).

Podemos então alterar o arquivo SIST.BAT deixando-o assim:

```
%1 CADASTR COPCADS
%1 MOVT COPMOVT
DIGIT
VERIFIC
ATUALI
LISTAG
```

Nas duas primeiras linhas, os sinais %1 serão substituidos pelos comandos apropriados na hora que forem executados. Quando quisermos rodar o sistema fazendo uma cópia dos arquivos, devemos digitar:

```
A>SIST COPY
```

Ao receber esta linha, o HB-DOS substituirá os %1 que forem encontrados no arquivo SIST pela palavra COPY. Esta substituição é feita apenas na hora de execução. Assim, serão executados os seguintes comandos:

```
COPY CADASTR COPCADS
COPY MOVTO COPMOV
DIGIT
VERIFIC
ATUALI
LISTAG
```

Se for digitado:

```
A>SIST REM
```

o HB-DOS executará os seguintes comandos:

```
REM    CADASTR    COPCADS
REM    MOUTO      COPMOU
DIGIT
VERIFIC
ATUALI
LISTAG
```

O processamento em lote no HB-DOS é semelhante ao do HB-MCP, porém, no HB-DOS ele já existe no próprio sistema operacional, enquanto que no HB-MCP é necessário utilizar um programa à parte (o SUBMIT.COM). O resultado final, entretanto, é o mesmo!

O símbolo %0 será sempre substituído pelo próprio nome do arquivo em que ele está presente, não sendo possível, portanto, atribuir-lhe outros comandos.

PECULIARIDADES DO HB-DOS

Quando o HB-DOS foi desenvolvido, um dos objetivos perseguidos foi a semi-compatibização dele com o MS-DOS. Assim, os arquivos gravados com o MS-DOS por outros micros (como o IBM-PC, por exemplo), podem ser acessados pelo HB-DOS.

Outro ponto importante a salientar é o fato de, apesar de o comando TIME existir no HB-DOS, seu uso não é possível, pois o HOTBIT não possui relógio interno. Sempre que se comandar TIME a mensagem "Não tem relógio" será apresentada.

Quando se termina uma linha teclando RETURN, ela é armazenada numa região da memória (buffer do teclado). Um dos recursos mais úteis do HB-DOS é a existência de teclas especiais para edição dos comandos contidos nesse buffer. Existem também teclas para controle da impressora.

TECLAS	FUNÇÃO
▶ ou ^\	Copia o próximo caractere do buffer para a tela, na posição do cursor.
SELECTn ou ^Xn	Copia todos os caracteres restantes no buffer até a achar o primeiro caractere n. Se n não estiver especificado, nada será copiado!
▼ ou ^_	Copia todos os caracteres restantes no buffer para a tela, a partir da posição do cursor.
DEL	Pula o caractere do buffer sob o cursor, deslocando os caracteres à sua direita uma posição para a esquerda, sem mudar a linha atual.
CLSn ou ^Ln	Pula todos os caracteres do buffer anteriores à primeira ocorrência do caractere n. Se n não estiver no buffer, não pula nada.
▲ ou ESC ou ^^ ou ^U ou ^[	Limpa a linha digitada na tela, mantendo o buffer inalterado.
◀ ou ◀◀ ou ^H ou ^]	Apaga da tela um caractere à esquerda do cursor, mantendo o buffer inalterado.
INS ou ^R	Ativa ou desativa a inserção de caracteres sob o cursor na tela e no buffer.
HOME ou ^K	Insere no buffer a linha que está na tela.
^P	Aciona o "eco" na impressora, fazendo com que tudo que for mandado para o vídeo vá, também, para a impressora.
^N	Desativa o "eco" na impressora.
^S	Interrompe a saída de caracteres no vídeo. Ao se pressionar uma tecla, a saída é retomada.
^C	Cancela a execução de um comando.

Uma última e importante observação sobre o HB-DOS é o fato se poder economizar memória, dependendo do uso que se fará do sistema. Ao carregar o HB-DOS (ou o DSK-BASIC!), se a tecla CTRL for mantida pressionada, todos os drives lógicos serão desconsiderados. O espaço alocado para eles na RAM será portanto, liberado para dados! Se a tecla SHIFT for mantida pressionada, o sitema não será carregado e o BASIC normal do HOTBIT será acionado.

CAPÍTULO V



DICIONÁRIO DE COMANDOS E FUNÇÕES

A seguir, encontram-se três dicionários de comandos e funções. O primeiro é referente ao DSK-BASIC, o segundo, é sobre o HB-MCP e, finalmente, o terceiro, é sobre o HB-DOS.

Em cada um deles, as funções e os comandos estão ordenados alfabeticamente. Se, por exemplo, você quiser saber para que serve o FORMAT, procure-o inicialmente no dicionário do DSK-BASIC. Depois, procure também nos outros dois dicionários, pois FORMAT é um comando presente nos três sistemas operacionais!

Certos comandos só existem num dos dicionários. Por exemplo, se você quiser saber como funciona o PIP, ao fazer a procura no dicionário do DSK-BASIC ou do HB-DOS, não o encontrará, pois esse é um comando apenas do HB-MCP.

Para todos os comandos e funções, foi usada uma notação convencional para indicar se algum parâmetro é optativo ou obrigatório e em que situações isso ocorre:

[parâmetro] ---> Os colchetes indicam que o parâmetro é opcional, podendo ser eventualmente omitido.

parâm1 ¦ parâm2 ---> A barra vertical indica que apenas um dos parâmetros que a ladeiam deve ser explicitado.

{ parâmetros } ---> As chaves indicam que ao menos um dos parâmetros internos a elas deve ser especificado.

A seguir, você pode observar o esquema geral de apresentação das informações para cada palavra dos dicionários.

título do comando ou função → 0000000

sistema operacional → 00000

Formato:	sintaxe do comando ou função.
Objetivo:	tarefa que o comando ou função executa.
Comentários:	observaçõs sobre o uso do comando ou função.
Exemplos:	exemplos de utilização do comando ou função.

BLOAD

Formato: BLOAD "{[disp] arquivo }"[,R¦,S][,deslocamento]

Objetivo: Carregar um programa em linguagem de máquina ou dados em binário do cassete ou disco para a memória.

Comentários: disp é o dispositivo no qual o programa será lido podendo ser CAS: (para o cassete), A: (para o drive A), e B: (para o drive B). Se for omitido, será assumido o drive corrente.

arquivo é o nome do programa que será lido e pode ser omitido se a leitura for efetuada do cassete.

Se o deslocamento for omitido, o programa será carregado no local indicado pelo comando BSAVE. Se não, o programa será carregado no endereço de memória especificado pelo BSAVE somado ao deslocamento.

Se for dada a opção ,R o programa será automaticamente executado a partir do endereço especificado em BSAVE, quando for carregado.

Se for dada a opção ,S será carregada a imagem de vídeo salva por BSAVE, na VRAM.

Exemplos:

```
BLOAD "CAS:",R
```

Carrega um programa do cassete e o executa assim que terminar a leitura.

```
BLOAD "A:PAISAG",S
```

Lê no drive A e carrega uma imagem de vídeo na VRAM.

```
BLOAD "B:JOGO1",R,-500
```

Lê o programa JOGO1 do drive B e o insere deslocando-o de 500 bytes em relação ao endereço original que foi especificado pelo BSAVE, executando-o logo a seguir.

Formato: BSAVE "{[disp] arquivo }",inic,fim[,exec!,S]

Objetivo: Gravar um programa em linguagem de máquina ou uma imagem de vídeo no gravador cassete ou no disco.

Comentários: disp é o dispositivo no qual o programa será gravado, podendo ser CAS: (para o cassete), A: (para o drive A), e B: (para o drive B). Se for omitido, será assumido o drive corrente.

arquivo é o nome com o qual será gravado o programa.

inic é o endereço inicial do bloco de memória a ser gravado.

fim é o endereço final do bloco de memória a ser gravado .

exec é o endereço de execução do programa. Se for omitido, será assumido o endereço de início do bloco.

A opção ,S indica que será gravado um bloco da memória de vídeo (VRAM). Ela possibilita a gravação de imagens de vídeo.

Exemplos:

```
BSAVE "CAS:JOGO",&H8000,&HBFFF,&H8010
```

Será gravado em fita cassete com o nome "JOGO" o programa em linguagem de máquina desde o endereço de memória &H8000 até &HBFFF, sendo executado a partir do endereço &H8010 (se utilizada a opção ,R do comando BLOAD).

```
BSAVE "A:FIGURA",&H0,&H3FFF,S
```

Será gravada no drive A a imagem do vídeo a partir do endereço &H0 e indo até o endereço &H3FFF da VRAM, com o nome FIGURA.

CLOSE

Formato: CLOSE [#arq1][,#arq2][,...]

Objetivo: Finalizar as operações de acesso a arquivos.

Comentários: arqn é o número do arquivo que foi usado no comando OPEN.

Se não for especificado nenhum número de arquivo, todos os arquivos abertos serão fechados.

Se houver dados no buffer do arquivo que ainda não foram gravados, então serão gravados e depois o arquivo será fechado.

Os comandos END, CLEAR, RUN, NEW e MAXFILES fecham os arquivos que estiverem abertos.

O comando STOP não fecha os arquivos.

Exemplos:

```
10 OPEN"A:TESTE" FOR OUTPUT AS # 1
20 PRINT#1,"MSX"
30 CLOSE#1
```

```
 5 MAXFILES=2
10 OPEN"A:ENTRA" FOR INPUT AS # 1
20 OPEN"A:SAI" FOR OUTPUT AS # 2
30 INPUT#1,A$
40 PRINT#2,A$
50 CLOSE#1
60 PRINT#2,"FIM"
70 CLOSE# 2
```

Formato: COPY "[disp1]arq1" TO "{[disp2][arq2]}"

Objetivo: Fazer cópias de um ou mais arquivos em discos.

Comentários: disp1 é o nome do dispositivo em que está o arquivo a ser copiado.

disp2 é o nome do dispositivo em que será feita a cópia.

arq1 é o nome do arquivo que será copiado.

arq2 é o nome do arquivo cópia, e pode ter nome diferente de arq1.

Mesmo tendo apenas um drive, pode-se operar com ele como se existissem dois drives (A: e B:). Nesse caso, é possível fazer a cópia de um arquivo de um disco para outro. O DSK-BASIC solicitará a troca dos discos, quando necessário, aguardando que você digite qualquer tecla (menos CTRL+STOP).

Pode-se usar o "*" e o "?" como nome de arq1 para copiar vários arquivos de uma só vez.

Na ausência do nome do arq2 será assumido o mesmo nome do arq1.

Exemplos:

```
COPY"A:ARQ.BAS" TO "A:ARQ.COP"

COPY"A:PROG1" TO "B:"

COPY"A:*.*" TO "B:"
```

CVI, CVS, CVD

Formato: variável numérica 2 = CVI(variável string 2)
variável numérica 4 = CVS(variável string 4)
variável numérica 8 = CVD(variável string 8)

Objetivo: Converter variáveis strings em variáveis numéricas.

Comentários: variável numérica 2 é uma variável inteira.

variável numérica 4 é uma variável de precisão simples.

variável numérica 8 é uma variável de precisão dupla.

variável string 2 é uma variável string com 2 bytes.

variável string 4 é uma variável string com 4 bytes.

variável string 8 é uma variável string com 8 bytes.

Todos os valores numéricos a serem gravados nos arquivos randômicos devem ser convertidos para variáveis string pelas funções MKI$, MKS$ e MKD$ e desconvertidos para valores numéricos novamente após a leitura do registro,pelas funções CVI, CVS e CVD.

Exemplos:

```
10 OPEN "ARQUIVO" AS #1 LEN = 14
20 FIELD #1,2 AS BI$,4 AS BS$,8 AS BD$
30 LSET BI$ = MKI$(4)
40 LSET BS$ = MKS$(12)
50 LSET BD$ = MKD$(22)
60 PUT #1,1
70 GET #1,1
80 PRINT CVI(BI$)
90 X = CVS(BS$)
100 PRINT 2 * CVD(BD$)
110 CLOSE
120 END
```

Formato: DSKF(número do drive)

Objetivo: Obter o número de blocos lógicos livres no disco.

Comentários: número do drive é o número do drive a se obter sendo 0 para o drive corrente, 1 para o drive A, 2 para o drive B, e assim por diante.

Cada bloco de disco pode armazenar 1024 bytes (1 Kbyte).

Exemplos:

```
PRINT DSKF(0)

PRINT DSKF(1)
```

Formato: DSKI$(número do drive,número do setor)

Objetivo: Ler um setor do disco.

Comentários: número do drive é o número do drive a ser lido, sendo 0 para o drive corrente, 1 para o drive A, 2 para o drive B e assim por diante.

número do setor é o número do setor lógico a ser lido, começando do setor 0.

Esta função retorna uma string nula.

Os endereços &HF351 e &HF352 contém o endereço de memória onde foram armazenados os dados.

Nenhuma verificação é feita pelo sistema para checar se o número do setor é valido ou não.

A área da memória apontada por &HF351 e &HF352 é destruída quando outro comando para disco, como FILES, OPEN, CLOSE, PRINT #, etc, é executada.

Exemplos:

```
10  CLEAR 1000
20  INPUT "LER QUAL SETOR ";X
30  A$=DSKI$(0,X)
40  ED=PEEK(&HF351)+256*PEEK(&HF352)
50  FOR I = 0 TO 255
60  PRINTHEX$(I+ED),HEX$(PEEK(I+ED))
70  NEXT I
80  GOTO 20
```

Formato: DSKO$ número do drive,número do setor

Objetivo: Gravar dados diretamente em um setor do disco.

Comentários: As posições de memória &HF351 e &HF352 apontam para a região da memória onde se localizam os dados a serem gravados.

número do drive = 0 significa drive corrente;
número do drive = 1 significa drive A ;
número do drive = 2 significa drive B ;
etc...

número do setor é o número do setor lógico onde serão gravados os dados. O primeiro setor do disco é o setor 0.

Qualquer comando de acesso a disco (OPEN,CLOSE, FILES, etc.) modifica os dados apontados pelos endereços &HF351 e &HF352.

Muito cuidado ao utilizar este comando, pois ele pode inutilizar os dados presentes no disco.

Exemplos:

```
10 ED=&H9000
20 HI=INT(ED/256)
30 LO=ED-256*HI
40 POKE &HF351,LO
50 POKE &HF352,HI
60 DSKO$ 0,0
```

Formato: EOF(número do arquivo)

Objetivo: Indicar, se for encontrado, o fim de um arquivo.

Comentários: número do arquivo é o número com o qual o arquivo foi aberto no comando OPEN.

EOF só pode ser utilizado com arquivos sequenciais abertos como entrada.

Ao ser encontrado o fim do arquivo, EOF retorna com -1, caso contrário com 0.

Deve-se usar EOF antes de cada comando INPUT # para se evitar uma mensagem de erro.

Exemplos:

```
10  OPEN "DADOS" FOR INPUT AS #1
20  IF EOF(1) THEN CLOSE : END
30  INPUT #1,X$
40  PRINT X$
50  GOTO 20
```

Formato: FIELD[#]nº arq,nº de bytes AS var1 [,nºde bytes AS var2][,nºde bytes AS var3][,...]

Objetivo: Alocar espaço para variáveis strings no buffer para os arquivos randômicos.

Comentários: Antes de se usar o PUT ou o GET é necessário utilizar FIELD para formatar o buffer do disco.

nº arq é o número utilizado na abertura do arquivo com o comando OPEN.

nºde bytes indica o tamanho que terá, dentro do buffer, cada variável string var1, var2, etc...

A soma de todos os números de bytes não deve ser maior que o valor especificado por LEN no comando OPEN, senão o DSK-BASIC emitirá uma mensagem de erro.

Nunca utilize variáveis strings usadas em FIELD em comandos como INPUT e LET , senão elas deixarão de fazer parte do buffer do disco.

Exemplos:

```
10 OPEN"A:RANDOC" AS #1 LEN=15
20 FIELD #1,10 AS A$,2 AS B$, 3 AS C$
```

Neste exemplo, foram alocados 10 bytes para a variável A$, 2 para B$ e 3 para C$.

```
10 OPEN"A:ARQ1" AS #1 LEN=20
20 FIELD#1,10 AS C1$,10 AS C2$
30 FIELD#1,5  AS A1$,15 AS A2$
```

Neste exemplo, tanto o comando FIELD da linha 20 como o comando FIELD da linha 30 terão efeito. Neste caso, os dados recebidos pelo comando GET estarão presentes tanto em C1$ e C2$ quanto em A1$ e A2$.

FILES

Formato: FILES ["[disp][nome do arquivo]"]

Objetivo: Mostrar os nomes dos arquivos gravados no disco.

Comentários: disp é o nome do dispositivo que será usado e pode ser apenas A: (para o drive A) ou B: (para o drive B).

Se não for especificado o nome do arquivo, serão mostrados todos os arquivos presentes no drive corrente.

Se for especificado o nome do arquivo, apenas ele será mostrado.

Pode-se usar * e ? como nome do arquivo.

No caso de não ser encontrado o nome do arquivo, será emitida uma mensagem de erro.

Exemplos:

```
FILES "A:*.BAS"
```

Mostra todos os arquivos do drive lógico A: cujos nomes terminam com o sufixo .BAS.

```
FILES
```

Mostra todos os arquivos do drive corrente.

```
FILES "B:"
```

Mostra todos os arquivos do drive lógico B: .

```
FILES "B:TES??.BAS"
```

Mostra todos os arquivos do drive lógico B: cujos nomes terminam com o sufixo .BAS e, simultaneamente, começam por TES e têm cinco caracteres.

Formato: CALL FORMAT
_FORMAT

Objetivo: Formatar o disco para uso no DSK-BASIC.

Comentários: Ao ser dado este comando são feitas algumas perguntas ao usuário antes que a formatação seja iniciada. Isso é necessário para que o usuádefina o drive lógico (A: ou B:), o tipo do drive (3,5" ou 5,25") e se ele é face simples ou face dupla.

A formatação é necessária quando os discos são virgens, pois só assim se pode usá-los.

Após a formatação, todos os dados que eventualmente estavam no disco são perdidos.

Exemplos: CALL FORMAT

Formata o disco no drive a ser especificado.

Fomato: GET[#]número do arquivo[,número do registro]

Objetivo: Efetuar a leitura de um registro de arquivo randômico.

Comentário: número do arquivo é o número do arquivo utilizado no comando OPEN.

número do registro é o número do registro que se deseja ler. Se omitido, será lido o próximo registro.

O número do registro deve estar compreendido entre 1 e 4.294.967.295.

Após o comando GET, os dados estarão disponíveis nas variáveis strings do comando FIELD.

Exemplos:

```
10 OPEN"A:ARQUIVO" AS #1 LEN=10
20 FIELD#1,10 AS A$
30 GET #1,1
40 PRINT A$
50 CLOSE
```

Formato: INPUT# número do arquivo,var1[,var2][,...]

Objetivo: Ler dados de um arquivo.

Comentários: número do arquivo é o número com o qual o arquivo foi aberto com o comando OPEN.

var1,var2, var3, var4,... são as variáveis que receberão os dados.

Os dados deverão vir da mesma maneira em que viriam em um comando INPUT.

O ponto de interrogação não é mostrado na tela.

Para dados numéricos: os espaços em branco, retorno de carro (&H0D) e avanço de linha (&H0A) são ignorados. O primeiro caractere encontrado que não for um desses três será assumido como início dos dados. Os dados terminam com um desses caracteres ou também com a vírgula (,).

Dados alfanuméricos: Os três caracteres citados acima também são ignorados no início dos dados. Qualquer outro caractere será considerado como início dos dados. Se o caractere for aspas (") então o dado consistirá de todos os caracteres compreendidos entre esta aspas e a próxima que aparecer. Se o primeiro caractere não for aspas ("), então o dado terminará com o aparecimento de uma vírgula (,), retorno de carro (&H0D) ou avanço de linha (&H0A), ou após terem sido lidos 255 caracteres.

Ao encontrar o fim do arquivo (&H1A), é terminada a execução do comando INPUT#.

Exemplos:

```
10 OPEN"A:ARQUIVO" FOR INPUT AS #1
20 INPUT#1,A$
30 PRINT A$
40 CLOSE#1
```

INPUT$

Formato: INPUT$(nº de caracteres[,[#]nº de arquivo])

Objetivo: Ler uma sequência de caracteres de um arquivo.

Comentários: nº de caracteres indica quantos caracteres deverão ser lidos do arquivo.

nº de arquivo é o número com o qual o arquivo foi aberto no comando OPEN.

O arquivo deve ter sido aberto para entrada.

Se o número do arquivo não for especificado, os caracteres serão lidos do teclado e, nesse caso, não serão mostrados na tela.

Exemplos:

```
10  OPEN "DADOS" FOR INPUT AS #1
20  IF EOF(1) THEN CLOSE : END
30  X$ = INPUT$(1,#1)
40  PRINT HEX$(ASC(X$))
50  GOTO 20
```

Formato: KILL "[disp]nome do arquivo"

Objetivo: Apagar arquivos do disco.

Comentários: disp é o nome do dispositivo em que o arquivo a ser apagado se encontra. Pode ser apenas A: (no caso do drive A) ou B: (para o drive B).

nome do arquivo é o nome do arquivo que se deseja apagar

Não se pode apagar um arquivo que esteja aberto pelo comando OPEN, senão o DSK-BASIC emitirá uma mensagem de erro.

Pode-se usar o ponto de interrogação (?) e o asterisco (*) como nome do arquivo para apagar mais de um arquivo de uma vez.

Exemplos:

```
KILL "A:LIXO.BAS"
```

Apaga do drive A: o arquivo de nome LIXO.BAS.

```
KILL "*.BAS"
```

Apaga do drive corrente todos os arquivos terminados com .BAS .

Formato: LINE INPUT # número do arquivo,variável string

Objetivo: Ler uma linha completa (até no máximo 254 caracteres) de um arquivo sequencial para uma variável alfanumérica, ignorando delimitadores (exceto os que delimitam o registro).

Comentários: número do arquivo é o número no qual o arquivo foi aberto.

variável string é o nome da variável que receberá os dados do arquivo.

Este comando lê todos os caracteres do arquivo até que seja encontrada a sequência &H0D &H0A (delimitadores de fim de registro) e coloca-os todos na variável (exceto os delimitadores).

Este comando é útil quando se deseja ler um programa salvo em ASCII pelo comando SAVE.

Exemplos:

```
10 OPEN "A:PROG.BAS" FOR INPUT AS #1
20 LINE INPUT #1,A$
30 IF EOF(1) THEN CLOSE : END
40 PRINT A$
50 GOTO 20
```

Formato: LOAD "{[disp] nome do arquivo }"[,R]

Objetivo: Carregar um programa BASIC.

Comentários: disp é o nome do dispositivo onde o será lido o arquivo e pode ser A: (drive A), B: (drive B) ou CAS: (cassete).

nome do arquivo é o nome do programa a ser carregado.

O programa só será carregado se for programa BASIC.

Este comando se encarrega de saber se o programa foi gravado em ASCII ou não.

Se for utilizada a opção ,R o programa será automaticamente executado assim que for carregado.

O comando LOAD apaga totalmente qualquer programa que estiver na memória do micro. Assim que começa a carregar o novo programa, fecham-se todos os arquivos que estiverem abertos, apagando todas as variáveis.

No caso da opção ,R ser utilizada, os arquivos que estiverem abertos permanecerão abertos, permitindo assim que um programa seja quebrado em segmentos.

O programa que estiver na memória permanecerá intacto até que seja encontrado o novo programa a ser carregado e seja iniciada a tranferência.

Exemplos:

```
LOAD "PROG1.BAS"

LOAD "JOGUINHO.GAM",R
```

Formato: LOC(número do arquivo)

Objetivo: Devolver o número do último registro lido (ou gravado) em um arquivo randômico.

Comentários: número do arquivo é o número com o qual o arquivo foi aberto no comando OPEN.

Esta função só deve ser utilizada com arquivos randômicos.

Exemplos:

```
10 OPEN "DADOS" AS #1 LEN = 10
20 FIELD #1,10 AS A$
30 LSET A$ = "MSX"
40 PUT #1,1
50 PRINT LOC(1)
60 PUT #1,2
70 PRINT LOC(1)
80 GET #1,1
90 PRINT LOC(1)
100 CLOSE
110 END

RUN
 1
 2
 1
Ok
```

Formato: LOF(número do arquivo)

Objetivo: Obter o número de bytes gravados num arquivo.

Comentários: número do arquivo é o número com o qual o arquivo foi aberto no comando OPEN.

LOF pode ser usado tanto para arquivos sequenciais quanto para arquivos randômicos.

Exemplos:

```
10 OPEN "DADOS" FOR INPUT AS #1
20 PRINT "O ARQUIVO DADOS TEM ";
LOF(1);" BYTES GRAVADOS"
30 CLOSE
40 END
```

Formato: LSET variável string = expressão string

Objetivo: Mover dados para as variáveis do comando FIELD.

Comentários: variável string é a variável que receberá os dados e geralmente é uma variável definida no comando FIELD.

expressão string pode ser uma outra variável ou um texto qualquer.

LSET move os dados começando da esquerda para a direita, preenchendo com espaços em branco, se sobrar espaço, ou truncando os dados excedentes, a partir da direita, se faltar espaço.

As variáveis numéricas devem ser tranformadas em variáveis alfanuméricas antes de serem movidas (veja MKI$, MKS$ e MKD$).

LSET pode ser usado para variáveis que não estão no comando FIELD, permitindo um alinhamento à esquerda.

Exemplos:

```
LSET A$ = "ISTO E UM EXEMPLO"
LSET B$ = X$
A$ = SPACE$(30)
LSET A$ = "TEXTO A ESQUERDA"
```

Formato: MERGE "{[disp] nome do programa }"

Objetivo: Misturar um programa gravado em disco com outro programa já carregado na memória do computador.

Comentários: disp é o dispositivo em que se encontra o programa a ser carregado para a memória do micro.

nome do programa é o nome do programa que será misturado com o programa que já está na memória.

O programa do disco deverá, obrigatoriamente, ter sido gravado em ASCII, senão será emitida uma mensagem de erro.

Se o programa do disco contiver uma linha de número igual ao de uma linha do programa no micro, prevalecerá a linha proveniente do disco.

Após a execução do comando MERGE, o DSK-BASIC voltará com o "Ok", à espera de um comando.

Exemplos: MERGE "PROGR.ASC"

Sobrepõe ao programa na memória do micro o programa PROGR.ASC , que deve estar necessariamente gravado em formato ASCII.

MKI$, MKS$, MKD$

Formato: MKI$(valor inteiro)
MKS$(valor precisão simples)
MKD$(valor precisão dupla)

Objetivo: Converter valores numéricos em variáveis tipo string.

Comentários: valor inteiro pode ser número inteiro ou variável do tipo inteira.

valor precisão simples pode ser um número de precisão simples ou uma variável de precisão simples.

valor precisão dupla pode ser um número de precisão dupla ou uma variável de precisão dupla.

Todos os valores numéricos a serem gravados em um arquivo randômico devem ser convertidos para string antes de ser usado LSET ou RSET.

MKI$ transforma um número em uma string de 2 bytes

MKS$ transforma um número em uma string de 4 bytes

MKD$ transforma um número em uma string de 8 bytes.

Exemplos:

```
10 OPEN "DADOS" AS #1 LEN = 14
20 FIELD #1, 2 AS VI$,4 AS VS$, 8 AS
VD$
30 LSET VI$ = MKI$(2)
40 LSET VS$ = MKS$(X! + 5)
50 LSET VD$ = MKD$(Y#)
60 PUT #1,1
70 CLOSE
80 END
```

Formato: NAME "velho nome" AS "novo nome"

Objetivo: Modificar o nome de um arquivo.

Comentários: velho nome é o nome atual do arquivo.

novo nome é o nome com o qual irá ficar o ar- arquivo.

Deve ser utilizado o mesmo drive nos dois parâmetros.

Exemplos: NAME "ARQ1.BAS" AS "ARQOK.BAS"

Muda o nome do arquivo ARQ1.BAS para ARQOK.BAS.

Formato: OPEN"{[disp] arquivo }" [FOR modo] AS [#] nº [LEN=tamanho]

Objetivo: Abrir um arquivo para que ele possa ser usado.

Comentários: disp é o nome do dispositivo em que será aberto arquivo.

arquivo é o nome do arquivo que se deseja usar.

modo é a especificação de entrada ou saída para o arquivo, podendo ser: FOR OUTPUT (para especificar arquivo sequencial como saída), FOR INPUT (para especificar arquivo sequencial como entrada), ou FOR APPEND (para especificar arquivo sequencial como saída). Se for usado FOR APPEND os dados serão gravados após o último dado presente no disco, sem destruir o que já estava gravado.

Se modo for omitido, será assumido arquivo randômico.

nº é o número com que o arquivo será aberto, podendo ir de 1 até o valor especificado no comando MAXFILES. Seu valor máximo é 15.

Todo comando de acesso ao arquivo deverá referenciar o nº para que o DSK-BASIC saiba qual o arquivo a utilizar.

tamanho indica o tamanho do regitro se o arquivo for randômico.

Se for utilizado FOR INPUT ou FOR APPEND em um arquivo inexistente, será enviada uma mensagem de erro.

Exemplos:

```
OPEN "A:TESTE.ASC" FOR OUTPUT AS #1

OPEN "B:ARQ1" AS #1 LEN=14

OPEN "CACA" FOR INPUT AS #1
```

No primeiro exemplo, foi aberto um arquivo sequencial como saída no drive A.

No segundo exemplo, foi aberto um arquivo randôdômico com registros de 14 bytes de tamanho.

No último exemplo, o arquivo CACA foi aberto como entrada.

Formato: PRINT #número,dados1[,dados2][,dados3][,...]

Objetivo: Gravar dados num arquivo aberto como saída.

Comentários: número é o número com o qual o arquivo foi aberto.

dados1 pode ser qualquer expressão, numérica ou string.

dados2,dados3,etc... são análogas a dados1, porém, são opcionais.

Toda a sintaxe utilizada no comando PRINT para o vídeo continua valendo para o PRINT no arquivo.

A especificação USING pode ser usada de forma análoga ao seu uso com o comando PRINT.

Da mesma maneira que os dados sairiam no vídeo com o comando PRINT, eles serão enviados ao disco.

Se os dados a serem enviados são numéricos, eles devem estar separados por ponto e vírgula (;).

Se os dados são alfa-numéricos, devem estar separados por vírgula (,).

Se algum dos dados alfa-numéricos a serem gravados contiver uma vírgula embutida, é melhor separar os dados através de aspas (") utilizando CHR$(34).

Exemplos:

```
PRINT #1,A;B;C

PRINT #1,A$,B$

PRINT #1,CHR$(34)A$CHR$(34)CHR$(34)B$
CHR$(34)
```

Formato: **PUT [#]número do arquivo[,número do registro]**

Objetivo: Gravar um registro no arquivo randômico.

Comentários: número do arquivo é o número com o qual o arquivo foi aberto.

número do registro indica qual registro receberá os dados, podendo ir de 1 até 4.294.967.295 . Se omitido, o próximo registro em relação ao último GET ou PUT será o assumido.
Os dados a serem gravados devem ter sido movidos previamente pelos comados LSET OU RSET.

Exemplos :

```
10 OPEN "ARQ" AS #1 LEN=10
20 FIELD #1,10 AS A$
30 LSET A$ = "EXEMPLO"
40 PUT #1,1
50 CLOSE
60 END
```

Formato: RSET variável string = expressão string

Objetivo: Mover dados para as variáveis do comando FIELD

Comentários: variável string é a variável que receberá os dados e geralmente é uma variável definida no comando FIELD.

expressão string pode ser uma outra variável ou um texto qualquer.

RSET move os dados começando da direita para a esquerda, preenchendo com espaços em branco, se sobrar espaço, ou truncando os dados excedentes, a partir da direita, se faltar espaço.

As variáveis numéricas devem ser tranformadas em variáveis alfanuméricas antes de serem movidas (veja MKI$, MKS$ e MKD$).

RSET pode ser usado para variáveis que não estão no comando FIELD, permitindo um alinhamento do pela direita.

Exemplos:

```
RSET A$ = "ISTO E UM EXEMPLO"
RSET B$ = X$

A$ = SPACE$(30)
RSET A$ = "TEXTO A DIREITA"
```

Formato: RUN [número da linha]
RUN "[disp]nome do programa"[,R]

Objetivo: Iniciar a execução de um programa BASIC.

Comentários: No primeiro formato, se for especificado o número da linha, então será executado o programa que está na memória a partir da linha especificada e, se não for especificado o número da linha, então será executado o programa que está na memória do micro a partir da primeira linha.

Se for especificado o número da linha e esta não existir, então será enviada uma mensagem de erro.

No segundo formato, será executado o programa indicado por nome do programa.

Se for especificada a opção ,R todos os arquivos abertos permanecerão abertos.

Se não for especificada a opção ,R então os arquivos abertos serão fechados e depois será carregado o programa.

Se não existir um programa com o nome dado será emitida uma mensagem de erro.

Todas as variáveis são apagadas ao ser executado RUN.

Exemplos:

```
RUN

RUN 10

RUN "PRGRAMA.BAS"

RUN "PROGRAMA.BAS",R
```

Formato: SAVE {"[disp] nome do programa "}[,A]

Objetivo: Gravar o programa que está na memória.

Comentários: disp é o dispositivo em que o arquivo deverá ser gravado.

nome do programa é o nome com o qual o programa será gravado.

Se for especificada a opção ,A o programa será gravado na forma ASCII, caso contrário será gravado na forma binária.

Exemplos: SAVE "PROGRAMA"

Grava no drive corrente e no formato binário o arquivo PROGRAMA.

SAVE "CAS:PROG"

Grava em fita cassete no formato ASCII o arquivo PROG.

SAVE "B:PROG.ASC",A

Grava no drive B: e em formato ASCII o arquivo PROG.ASC .

SYSTEM

Formato: CALL SYSTEM
_SYSTEM

Objetivo: Voltar ao HB-DOS.

Comentários: SYSTEM só funciona quando o DSK-BASIC for chamado pelo HB-DOS através do comando BASIC.

Todos os dados da memória são apagados quando este comando é executado.

Se este comando for usado sem que o DSK-BASIC ou o HB-DOS estejam com o controle do micro, será emitida uma mensagem de erro.

Exemplos: CALL SYSTEM

Formato: AUTOLD[comando]

Objetivo: Gravar no drive corrente um comando para ser executado automaticamente logo após o carregamento do sistema.

Comentários: comando é qualquer comando do HB-MCP.

Se comando for omitido, retira o auto-comando do disco.

Exemplos: A>AUTOLD DIR

O disquete no drive A: receberá o comando DIR para ser executado toda vez que se carregar o sistema desse disco.

B>AUTOLD

O disquete no drive B: não terá auto-comando.

BACKUP

Formato: BACKUP

Objetivo: Copiar um disquete.

Comentários: O programa pede para ser colocado o disco a ser copiado e o disco que receberá a cópia alternadamente durante o processo de cópia.

O disquete colocado no drive origem será copiado no disquete colocado no drive destino.

Exemplos: B>BACKUP

Formato: COPY [arquivo]

Objetivo: Copiar arquivos com um só drive.

Comentários: arquivo é o nome do arquivo a ser copiado.

O arquivo especificado será lido do disquete origem e copiado no disquete destino com a utilização do mesmo drive.

A ausência do nome de arquivo permite que todos os arquivos do disquete orígem sejam copiados no disquete destino.

Os caracteres "?" e "*" podem ser usados no nome do arquivo.

Os arquivos solicitados serão copiados enquanto houver espaço no disquete destino.

O programa COPY requisita os discos de origem e destino quando necessários.

Exemplos:

```
A> COPY BACKUP.COM
```

O arquivo BACKUP.COM será copiado.

```
A> COPY *.DAT
```

Todos arquivos com a extensão .DAT serão copiados.

```
A> COPY *.*
```

Todos arquivos do disquete serão copiados.

Formato: DIR [disp][nome do arquivo]

Objetivo: Mostrar o nome dos arquivos presentes no disco.

Comentários: disp pode ser apenas A: ou B: .

Se for fornecido apenas disp então serão mostrados todos os arquivos tipo DIR presentes no disco.

Se for fornecido o nome do arquivo, então será mostrado este arquivo, caso exista, ou uma mensagem de erro, caso ele não exista no disco.

Pode-se utilizar o ponto de interrogação (?) e o asterisco (*) no nome do arquivo.

Exemplos: A> DIR

Mostra todos os arquivos presentes no drive corrente.

A> DIR B:

Mostra todos os arquivos presentes no drive B .

A> DIR B:*.COM

Mostra todos os programas presentes no drive B .

Formato: DRINP

Objetivo: Configurar o dispositivo de entrada.

Comentários: Esse programa permite compatibilizar a entrada de informações do HOTBIT com diversos outros padrões.

O padrão adotado como "default" é o ABNT.

O programa apresenta um menu com as opções disponíveis.

Formato:	DROUT
Objetivo:	Configurar o dispositivo de saída.
Comentários:	Esse programa permite compatibilizar a saída de informações do HOTBIT com diversos outros padrões. O padrão adotado como "default" é o ABNT. O programa apresenta um menu com as opções disponíveis.

Formato: DSKCNV

Objetivo: Permitir o acesso do HB-MCP a discos com formatações diferentes.

Comentários: Este utilitário permite a utilização de disquetes com formatos de outros fabricantes, diferentes do utilizado pelo HB-MCP, que se utilizem de sistema operacional compatível com o CP/M.

O acesso será realizado através de um drive lógico E: .

Se um sistema possui dois drivers (logicamente denominados A: e B:) após a execução do utilitário DSKCNV, o drive B: ficará configurado na formatação selecionada. O sistema passará a possuir três drives lógicos (A:, B: e E:), sendo B: e E: o mesmo drive físico.

Se o sistema possuir apenas um drive físico, este será usado também como drive lógico E: .

O programa mostra um menu com as configurações possíveis.

Exemplos: A>DSKCNV

Formato: DUMP[disp]arquivo

Objetivo: Visualizar o conteúdo de um arquivo.

Comentários: disp é o dispositivo em que está o arquivo a ser lido.

disp é opcional. Se omitido, será assumido o drive corrente.

arquivo é o arquivo a ser visualizado.

Este utilitário permite ver como realmente o arquivo está gravado em disco, mostrando-o byte a byte junto com os caracteres correspondentes.

Exemplos: A>DUMP B:STAT.COM

Formato: ERA [disp]arquivo

Objetivo: Apagar um arquivo do disco.

Comentários: disp é o dispositivo (drive A: ou B:) em que se encontra o arquivo a ser apagado.

arquivo é nome do arquivo a ser eliminado.

Um arquivo apagado pelo comando ERA não pode mais ser acessado.

Pode-se apagar mais de um arquivo de uma vez utilizando o ponto de interrogação (?) ou o asterisco (*) no arquivo.

Se o arquivo a ser apagado não estiver no disco, então será emitida uma mensagem de erro.

Ao se usar *.* para apagar todos os arquivos de um disco, uma pergunta é feita para confirmar a operação.

Exemplos:

```
A>ERA A:PROG1.BAS
```

Apaga o programa PROG1.BAS do drive A.

```
A>ERA B:*.COM
```

Apaga todos os programas do drive B.

Formato: FORMAT drive

Objetivo: Iniciar disquetes para trabalho.

Comentários: O disquete no drive especificado será inicializado.

drive pode ser A: ou B:.

O conteúdo do disquete a ser formatado será perdido.

Exemplos : A>FORMAT B:

O disquete no drive B: será formatado.

A>FORMAT A:

O disquete no drive A: será formatado.

Formato: FUNCOES

Objetivo: Listar o conteúdo atual das teclas de funções.

Comentários: No HB-MCP as teclas de funções podem ser redefinidas. Para saber o que elas contém, basta usar o comando FUNCOES.

Exemplos: A>FUNCOES

Formato: PIP {[displ]arq-dest}=[disp2]arq-orig[/opções] [,[disp2]arq-orig]/opções ...

Objetivo: Copiar arquivos entre periféricos.

Comentários: displ é o dispositivo em que se será gerado um arquivo cópia.

disp2 é o dispositivo onde se encontra o arquivo a ser copiado.

arq-dest é o arquivo a ser gerado.

arq-orig é o arquivo original a ser copiado.

arq-dest pode ser diferente do arq-orig.

arq-dest pode ser omitido. Nesse caso receberá o mesmo nome do arq-orig.

Exemplos:

```
A>PIP B:=A:ARQ.BAS
```

Copia ARQ.BAS do drive A para o drive B.

```
A>PIP B:COPIA2=A:COP.COM
```

Copia COP.COM do drive A para o drive B mudando o nome para COPIA2.

```
A>PIP B:=A:*.COM
```

Copia todos os arquivos com a extensão .COM do drive A para o drive B.

```
A>PIP A:=B:*.*
```

Copia todos os os arquivos do drive B para o drive A.

```
A>PIP A:ARQ=B:ARQ1, B:ARQ2
```

Copia ARQ1 do drive B para o drive A com o nome ARQ e logo em seguida copia ARQ2 do drive B para o drive A juntando-o ao final do ARQ. Após a cópia, A:ARQ contém tanto ARQ1 quanto ARQ2, um logo após o outro.

```
A>PIP LST:=A:QQC.TXT
```

Copia na impressora o arquivo QQC.TXT que esta no drive A .

Opções do PIP durante a cópia:

A - Copia apenas os arquivos que sofreram alguma modificação.

Dn - Deleta os caracteres que passarem da coluna n.

E- Mostra no vídeo os caracteres que estão sendo copiados.

F - Filtra os caracteres &H0C (alimentacão de folha) do arquivo original.

Gn - Copia arquivos do usuário n .

H - Testa a validade dos arquivos gravados no formato padrão INTEL.

L - Transforma os caracteres em letras maiúsculas para letras minúsculas.

N - Numera as linhas copiadas no arquivo destino.

N2 - Numera as linhas copiadas no arquivo destino.

O - copia todo o arquivo mesmo encontrando o caracter &H1A que indica o seu final.

Pn - Pula de página a cada n linhas copiadas.

Qseq^Z - Termina a cópia quando encontra a sequência de caracteres seq no arquivo.

R - Copia os arquivos mesmo que sejam de sistema (SYS).

Sseq^Z: Começa a cópia apenas quando encontrar a sequência de caracteres seq dentro do arquivo.

Tn - Expande todos os caracteres TAB para uma coluna múltipla de n.

U - Transforma letras minúsculas em letras maiúsculas.

V - Verifica se a cópia foi bem feita.

W - Copia os arquivos mesmo estando protegidos (arquivos R/O).

Z - Zera o bit 7 de cada byte copiado.

Formato: REN [disp]novo nome=[disp]velho nome

Objetivo: Modificar o nome de um arquivo.

Comentários: disp é o dispositivo em que se encontra o arquivo.

novo nome é o nome que será atribuído ao arquivo.

velho nome é o nome atual do arquivo.

Se for especificado disp, então este deve ser o mesmo, tanto no novo nome quanto no velho nome.

Os dados dos arquivos não são alterados, apenas o nome é que é alterado.

Não se pode utilizar o ponto de interrogação (?) nem o asterisco (*) nos nomes dos arquivos.

Exemplos:

```
A>REN PROG1.BAS=PROG1.AUX
```

Muda o nome do arquivo PROG1.AUX para PROG1.BAS no drive corrente.

```
A>REN B:X.COM=X2.COM
```

Muda o nome do arquivo X2.COM do drive B para X.COM .

Formato: SAVE número de páginas [disp]arquivo

Objetivo: Copiar os dados contidos na memória do computador a partir do endereço &H0100 no arquivo especificado.

Comentários: número de páginas é o numero de blocos lógicos a serem gravados no arquivo especificado.

disp é o dispositivo em que se encontra o arquivo que receberá os dados.

arquivo é o arquivo que receberá os dados.

A região da memória gravada em disco inicia sempre em &H0100 (ou seja, o TPA).

O tamanho de uma página é de 256 bytes.

Exemplos:

```
A>SAVE 8 MEM.COM
```

Grava um arquivo de 2 Kbytes com o nome MEM.COM contendo os dados da memória desde o endereço &H0100 até &H0900.

```
A>SAVE 32 B:COPIAM.BAS
```

Grava no drive B um arquivo de 8 Kbytes com o nome COPIAM.BAS contendo os dados da memória desde o endereço &H0100 até &H2100.

Formato: STAT [parâmetro]

Objetivo: Informar o "status" dos discos e arquivos.

Comentários: parâmetro é uma especificação do tipo de informação a ser fornecida pelo STAT. Se omitido, o STAT fornece o espaço livre em disco e se ele está ou não protegido para gravação.

parâmetro pode ser : A: , B: , DEV: , VAL: , nome do arquivo , USR: , dispositivo lógico=dispositivo físico.

Exemplos:

```
A>STAT A:
```

Fornece o espaço livre no drive A.

```
A>STAT ARQ1
```

Fornece as características do arquivo ARQ1.

```
A>STAT DSK:
```

Mostra as características do drive corrente.

```
A>STAT DEV:
```

Mostra as associações entre dispositivos lógicos e físicos.

```
A>STAT VAL:
```

Mostra um menu do que o STAT pode fazer.

```
A>STAT TTY:=VD4:
```

Associar TTY: ao vídeo de 40 colunas.

Formato: SYSGEN

Objetivo: Copiar o HB-MCP para outro disco.

Comentários: O sistema operacional HB-MCP presente no drive corrente será copiado em outro disquete.

O programa SYSGEN solicita a troca do disco sempre que necessário.

Exemplos: A>SYSGEN

O sistema operacional HB-MCP presente no disco do drive A será gravado em um outro disco.

Formato: TERM

Objetivo: Configura caracteres de controle para o vídeo.

Comentários: Inicialmente o sistema está configurado com o padrão VT-52.

O programa TERM permite ao vídeo do HOTBIT aceitar comandos do padrão de vídeo ADM-3A.

O programa mostra um menu com as duas opções.

Formato: TYPE [disp]arquivo

Objetivo: Mostrar no vídeo (e na impressora, no caso de ter sido teclado ^P) o conteúdo de um arquivo.

Comentários: disp é o dispositivo em que se encontra o arquivo a ser pesquisado (drive A: ou B:).

arquivo é o arquivo a ser pesquisado.

O arquivo deve conter apenas caracteres de texto senão poderá dar a entender que o micro ficou "maluco" ao se utilizar o comando TYPE.

Pode-se teclar STOP (ou ^S) para congelar o vídeo durante a execução do TYPE.

Qualquer tecla encerra a execução do TYPE.

Exemplos:

```
A>TYPE A:ARQ.BAS
```

Mostra no vídeo os dados gravados no arquivo de nome ARQ.BAS do drive A.

```
A>TYPE B:TEXTO.ASC
```

Mostra os dados gravados no arquivo TEXTO.ASC presente no drive B.

Formato: USER número do usuário

Objetivo: Mudar o número do usuário corrente.

Comentários: número do usuário deve estar entre 0 e 15.

Quando o HB-MCP é inicializado é assumido USER 0

Exemplos: A>USER 10

Muda o número do usuário para 10.

10A>USER 0

Retorna para o usuário zero.

Formato: BASIC [programa]

Objetivo: Sair do HB-DOS e entrar no DSK-BASIC.

Comentários: programa deve ser necessariamente um programa em BASIC. Se ele for especificado, será carregado e executado automaticamente.

Para voltar ao HB-DOS, deve-se utilizar o comando CALL SYSTEM.

O cartão de 80 colunas (se existir) é desconsiderado quando o micro passa do HB-DOS para o DSK-BASIC, pois este não trata o cartão de 80 colunas.

Exemplos:

```
A>BASIC
```

Retorna do HB-DOS para o DSK-BASIC.

```
A>BASIC PROG1
```

Retorna ao DSK-BASIC e executa o programa PROG1.

Formato: COPY {[d1][a1]}[/o1][+ {[d2][a2]}[/o2]][+ ...]
[+ {[dn][an]}[/on]] {[dispd][arqd]}[/opd]

Objetivo: Copiar arquivos entre periféricos.

Comentários: d1, d2, d3, etc. são os dispositivos em que estão os arquivos a serem copiados.

a1, a2, a3, etc. são os arquivos a serem copiados.

o1, o2, o3, etc. são opções que definem os tipos dos arquivos ou especificações de verificação, e podem ser: A (para arquivos em ASCII), B (para arquivos em binário) ou V (para que seja feita a verificação).

dispd é o dispositivo de destino em que se encontra o arquivo que receberá a(s) cópia(s).

arqd é o arquivo destino onde serão feitas as cópias.

opd é do mesmo tipo que o1 (A,B ou V).

Se arqd for omitido, será assumido o mesmo nome do arquivo a ser copiado, desde que ele seja único.

Se existir apenas um drive, o comando COPY solicitará as trocas dos discos sempre que for necessário, podendo ser efetuadas cópias de arquivos de um disco a outro.

Com o COPY pode-se utilizar também o ponto de interrogação (?) e o asterisco (*).

Exemplos:

```
A>COPY A.MAC + B.MAC C.MAC
```

Junta A.MAC e B.MAC e os copia no arquivo C.MAC.

```
A>COPY *.COM B:
```

Copia todos os programas do drive A para o B .

```
A> COPY A.MAC LST
```

Lista o arquivo A.MAC na impressora.

Formato: DATE [dia-mês-ano]
DATE [dia/mês/ano]
DATE [dia,mês,ano]

Objetivos: Mostrar ou alterar a data do sistema.

Quando um arquivo é criado ou alterado, a data é gravada junto com o arquivo.

Se teclar somente DATE, a data corrente é mostrada, e o sistema fica esperando uma nova data. Não querendo alterar a data corrente, basta teclar RETURN.

São aceitos o sinal de menos (-), a barra (/) e a vírgula (,) como separadores.

A data fornecida deve obrigatoriamente ser uma uma data válida.

Formato: DEL [arquivo]

Objetivo: Apagar do diretório os arquivos especificados.

Comentários: arquivo é arquivo que se quer apagar.

Pode-se usar o ponto de interrogação (?) e o asterisco (*) para se apagar mais de um arquivo de uma só vez.

No caso de se teclar DEL *.* o sistema perguntará se realmente os arquivos devem ser apagados.

Veja também ERASE.

Exemplos:

```
A>DEL A:ARQ1.BAS
```

Apaga o arquivo ARQ1.BAS do drive A.

Formato: DIR [disp] [arquivo] [/P] [/W]

Objetivo: Mostrar os arquivos gravados no disco.

Comentários: disp é o dispositivo em que se quer procurar os arquivos e pode ser A: (para o drive A) e B: (para o drive B). Se omitido, será assumido o drive corrente.

arquivo é o arquivo que se quer procurar no dispositivo especificado. Se for omitido, todos arquivos encontrados serão listados.

Pode-se usar o ponto de interrogação (?) e o asterisco (*) no arquivo para selecionar alguns arquivos com características em comum.

Se for dada a opção /W somente os nomes dos arquivos serão mostrados.

Se for dada a opção /P o DIR mostrará os arquiquivos gravados até a tela ficar cheia, solicitando então que seja precionada alguma tecla para dar continuidade ao processo.

No caso de listar o diretório de um disco formato MS-DOS/PC-DOS, os arquivos correspondentes a sub-diretórios são mostrados com a mensagem "<DIR>" no lugar do número de bytes.

Exemplos: A>DIR

Mostra os arquivos presentes no drive corrente.

A>DIR B: /P

Mostra os arquivos do drive B de página em página.

Formato: ERASE [arquivo]

Objetivo: Apagar do diretório os arquivos especificados.

Comentários: arquivo é arquivo que se quer apagar.

Pode-se usar o ponto de interrogação (?) e o asterisco (*) para se apagar mais de um arquivo de uma só vez.

No caso de se teclar ERASE *.* , o sistema perguntará se realmente os arquivos devem ser apagados.

Veja também DEL.

Exemplos:

```
A>ERASE A:ARQ1.BAS
```

Apaga o arquivo ARQ1.BAS do drive A.

Formato: FORMAT

Objetivo: Formatar um disco para uso do HB-DOS.

Comentários: Além de iniciar diretório e as tabelas, grava na trilha 0, setor 0 o LOADER.

O usuário deve responder a algumas perguntas do FORMAT para especificação de drive, etc.

O HB-DOS consegue formatar disquetes de 3,5" e e 5,25" , face simples ou dupla.

A formatação utilizada pelo HB-DOS é a mesma utilizada pelo DSK-BASIC.

Exemplos: A>FORMAT

Formato: MODE largura

Objetivo: Selecionar o número máximo de caracteres por linha, para o vídeo.

Comentários: largura é o número de caracteres por linha do vídeo.

O valor assumido na carga do sistema é 40 (ou 80, caso exista cartão de 80 colunas instalado).

Se houver cartão de 80 colunas, a largura poderá variar de 1 a 80; caso contrário, de 1 a 40.

Exemplos: A>MODE 40

Faz com que o vídeo fique com 40 colunas.

Formato: PAUSE [comentários]

Objetivo: Suspender a execução de um arquivo tipo "BATCH" mostrando na tela uma mensagem.

Comentários: Quando um arquivo tipo "BATCH" está sendo executado e possui, como um de seus comando, o PAUSE, ao encontrá-lo o HB-CCP suspende a execução dos comandos e aguarda que o usuário pressione alguтecla. Teclando ^C a execução é definitivamente interrompida. Qualquer outra tecla, se pressionada, dará prosseguimento à execução.

Se for especificada alguma mensagem, ela será impressa na tela durante a suspensão da execução do arquivo BATCH.

REM

Formato: REM [comentários]

Objetivo: Mostrar comentários no vídeo.

Comentários: REM também pode ser utilizado em arquivos para processamento em lote (tipo BATCH).

Os separadores aceitos nos comentários são os seguintes: espaço, vírgula e tabulador.

Formato: REN [disp]velhonome novonome
RENAME [disp]velhonome novonome

Objetivo: Mudar o nome de um arquivo.

Comentários: disp é o dispositivo em que se encontra o arquivo cujo nome será mudado e pode ser A: (para o drive A) ou B: (para o drive B). Se omitido, será assumido o drive corrente.

velhonome é o nome atual do arquivo.

novonome é o nome com o qual ele deverá ficar.

Se no drive especificado já existir um arquivo com o nome novonome, o HB-DOS acusará erro e não trocará o nome.

O ponto de interrogação (?) e o asterisco (*) podem ser usados nos nomes dos arquivos. Se aparecerem no velhonome, vão indicar os arquivos cujos nomes serão trocados. Quando são usados no novonome, os caracteres correspondentes às posições dos mesmos não sofrem alteração.

Exemplos:

```
REN A:BABA.BAS CACA.BAS
```

Altera o nome do arquivo BABA.BAS para CACA.BAS no drive A.

Formato:	TIME [horas:minutos:segundos]
Objetivo:	Mostrar e/ou alterar o instante marcado pelo relógio interno do micro.
Comentários:	TIME só funciona se houver relógio no micro.

Quando há relógio, a hora é gravada sempre que um arquivo é criado ou alterado.

Se o comando TIME é dado sem argumentos o HB-DOS mostra a hora corrente, incluindo centésimos de segundos, e espera nova hora do usuário. Caso não se deseje alterar a hora, basta digitar a RETURN.

O HOTBIT não possui relógio interno.

Os valores válidos de são:

horas : 0-23
minutos : 0-59
segundos : 0-59

O HB-DOS testa a consistência dos dados fornecidos.

Formato: TYPE [disp]arquivo

Objetivo: Mostrar o conteúdo de um arquivo ASCII na tela.

Comentários: disp é o dispositivo em que se encontra o arquivo a ser pesquisado e pode ser A: (drive A) ou B: (drive B). Se omitido, será assumido o drive corrente.

arquivo é o arquivo a ser pesquisado e deve estar no formato ASCII.

TYPE pode ser usado apenas para arquivos ASCII.

Pode-se usar ^P para mostrar também na impressora e ^S para suspender o "display" na tela.

Formato: VERIFY {[ON]|[OFF]}

Objetivo: Ligar ou desligar o modo verify.

Comentários: No modo verify, os dados após serem escritos no disco são lidos e verificados. Caso não confiram coloca mensagem de erro.

O default é VERIFY OFF, isto é, se não for usado o comando VERIFY ON, o sistema não aciona a verificação durante o acesso aos discos.

Usando o modo verify a escrita é muito mais confiável, todavia o tempo gasto na escrita se torna maior.

Na versão 1.0 do HB-DOS o modo verify não está implementado.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1	┼	┴	┬	┤	├	┼	│	─	┌	┐	└	┘	╳	╱	╲	┼
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	⌂
8	Ç	ü	é	â	Á	à	¨	ç	ê	í	ó	ú	Â	Ê	Ô	À
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
B	Ã	ã	Ĩ	ĩ	Õ	õ	Ũ	ũ	Ĳ	ĳ	¾	∽	◊	‰	¶	§
C	▂	▚	▆	▔	▬	▇	▎	▞	▊	▕	▉	▨	▧	▼	▲	▶
D	◀	⧗	⧓	▘	▗	▝	▖	▒	Δ	‡	ω	█	▄	▌	▐	▀
E	α	β	Γ	π	Σ	σ	µ	γ	Φ	θ	Ω	δ	∞	φ	∈	∩
F	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	▮	

```
1 SCREEN 1:INPUT"QUAL O CóDIGO EM HEXADE
CIMAL (LINHA & COLUNA) ";C$:PRINT:CH=VAL
("&H"+C$):PRINT"HEXADECIMAL=";C$:PRINT:P
RINT"DECIMAL="CH:PRINT:PRINT"APERTE RETU
RN PARA VÊ-LO","AMPLIADO E DEPOIS RETURN
 PARAPEDIR  OUTRO"
2 INPUT T$:SCREEN3:OPEN"GRP:"FOR OUTPUT
AS#1:PRESET(60,60):PRINT#1,C$;"=";:IF CH
>31 THEN PRINT#1,CHR$(CH)
3 IF CH<32 THEN PRINT#1,CHR$(1)+CHR$(64+
CH)
4 IF INKEY$=""THEN 4 ELSE RUN
```

APÊNDICE I

CÓDIGOS ASCII

O código ASCII (American Standard Code for Interchange Information) é um padrão internacional adotado em quase todos os microcomputadores. Na tabela a seguir, estão relacionados seus 128 caracteres com o nome correspondente.

Para obter seu código em hexadecimal, leia primeiro a linha e depois a coluna. Por exemplo, o caractere CONTROL Z (^Z) tem código hexadecimal 1A.

	A	
1	^Z SUB	caractere CONTROL Z , de código 1A (HEXADECIMAL) nome ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	^@ NUL	^A SOH	^B STX	^C ETX	^D EOT	^E ENQ	^F ACK	^G BEL	^H BS	^I HT	^J LF	^K VT	^L FF	^M CR	^N SO	^O SI
1	^P DLE	^Q DC1	^R DC2	^S DC3	^T DC4	^U NAK	^V SYN	^W ETB	^X CAN	^Y EM	^Z SUB	^[ESC	^\ FS	^] GS	^^ RS	^_ US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Na tabela da página ao lado, estão todos os 256 caracteres do HOTBIT. Para obter o código hexadecimal de um deles, use a mesma técnica da tabela acima. Querendo ver o caractere ampliado, digite o programa listado após a tabela, respeitando os espaços em branco.

APÊNDICE II

COMO INSTALAR O DRIVE

Ao instalar o disk drive no seu HOTBIT, alguns cuidados necessitam ser tomados de modo a garantir seu perfeito funcionamento. Dentre eles, salientamos os mais importantes:

- Certifique-se de que a voltagem da rede elétrica está de acordo com aquela indicada no equipamento: 110 ou 220 Volts, fazendo a mudança caso necessite.

- NUNCA insira o cartucho da interface do disk drive com o micro ligado, nem tampouco com o disk drive ligado.

- Quando utilizar o computador, ligue primeiro o disk drive e só depois, o micro. Ou então ligue o disk drive na tomada que existe na parte traseira do micro.

- Insira a interface em qualquer um dos SLOTS disponíveis no micro, pois o disk drive funciona perfeitamente em qualquer um deles.

- Insira o disquete com a face da etiqueta voltada para cima e de modo que parte em que ela está, seja a última a entrar no disk drive.

- Tome cuidado para que nem o disk drive nem os discos fiquem próximos de campos magnéticos, pois isso poderá danificá-los.

- Nunca introduza objetos (a não ser os disquetes!) dentro do disk drive.

- Desligue sempre primeiro o micro e, só depois, o drive.

- Nunca ligue ou desligue o drive com disquete dentro.

- Leia atentamente as instruções que aconpanham o disk drive e, somente depois de compreendê-las, comece a instalá-lo.

APÊNDICE III

MENSAGENS DE ERRO

Todos os sistemas operacionais esperam receber os comandos de uma certa forma e com uma certa sintaxe. Quando isso não ocorre, uma mensagem de erro é emitida informando que o comando não foi entendido. Nesse caso, normalmente bastará redigitar o comando de maneira correta. Há, porém, situações causadas por discos danificados,periféricos desligados, etc, nas quais não temos como corrigir o erro facilmente. De qualquer forma,uma mensagem será enviada para tomarmos conhecimento do erro e, se for possível, corrigí-lo. As mensagens de erro do DSK-BASIC, HB-MCP e do HB-DOS estão descritas a seguir.

MENSAGENS DO DSK-BASIC

Código de Erro

Arquivo aberto 54

Um arquivo foi aberto como OUTPUT para um arquivo já aberto ou KILL foi executado para arquivo aberto.

Arquivo ainda aberto 64

O arquivo não foi fechado.

Arquivo já existe 65

Já existe no disco um arquivo com o mesmo nome do arquivo dado na declaração NAME.

Arquivo não existe 53

Uma declaração LOAD, KILL ou OPEN foi executada para um arquivo que não existe no disco.

Arquivo sequencial 58

Uma declaração GET ou PUT foi executada para arquivo sequencial.

Campo maior 50

A declaração FIELD tentou alocar mais bytes que o especificado na opção tamanho do registro do arquivo randômico.

Comando direto no arquivo 57

Uma declaração direta foi detetada durante a carga de um programa no formato ASCII. Neste caso, o processo de carga é interrompido imediatamente.

Diretório cheio 67

Não existe mais espaço no diretório para criar uma nova entrada usando SAVE ou OPEN.

Disco cheio 66

Toda a área disponível no disco foi preenchida.

Disco protegido 68

Uma declaração PUT ou PRINT# foi executada para um disco protegido para escrita.

Drive errado 62

Foi fornecido um nome errado de drive.

Erro de disco 60

O disco não foi inicializado (formatado).

Erro interno 51

Um mau funcionamento interno ocorreu no DSK-BASIC. Entre em contato com a assistência técnica de seu drive ou micro.

Erro I/O 69

Durante um processo de leitura/escrita no disco, aconteceu um erro irreversível, isto é, o sistema não conseguiu recuperar.

Erro no setor 63

Foi detado um erro no setor do disco.

Falta disco 70

Não encontrou disco no drive especificado.

Falta OPEN 59

Uma declaração INPUT ou OUTPUT foi executada para um arquivo não aberto.

Fim do arquivo 55

Uma declaração INPUT foi executada para um arquivo vazio ou para um arquivo que já chegou ao fim. Para evitar esse erro, use a função EOF para detetar fim de arquivo.

Modo errado 61

Uma tentativa de executar a declaração PUT, GET ou LOF para um arquivo sequencial ou abrir o arquivo de um modo que não

seja FOR INPUT, FOR OUTPUT ou FOR APPEND ou default (randômico).

Modo inválido 71

O RENAME foi executado entre dois drives.

Nome do arquivo 56

Uma forma imprópria do nome do arquivo (pode ser um caracter) foi usado para as declarações LOAD, SAVE, KILL ou OPEN.

Número do arquivo 52

Um comando ou declaração foi executado para um número de arquivo que não foi aberto ou foi aberto com um outro número.

MENSAGENS DE ERRO DO HB-MCP

Erro de carga

Quando um arquivo .COM é muito grande para ser carregado na memória e executado.

Erro no drive A: R/O
^C Cancela, RETURN Ignora R/O

Disquete no drive foi colocado sem executar ^C para inicializá-lo. Troca inválida de disquetes.

Erro no drive A: Arquivo R/O
^C Cancela, RETURN Ignora R/O

Arquivo protegido contra escrita.

Erro no drive A: Setor Ruim
^C Cancela, RETURN repete,
Ignora com qualquer tecla

Ocorre quando o disquete está com problema em um setor ou não existe disquete no drive.

Erro no drive A: Seleção
Aperte uma tecla

Tentativa de referenciar um disco que não existe no sistema.

Já Existe

Tentativa de mudança de nome de um arquivo para um nome já existente (comando REN)

Não leu

Quando ocorrer erro de leitura de um arquivo dentro do comando TYPE

Nenhum arquivo

Surge quando um arquivo especificado não é encontrado.

Sem espaço

Quando não há espaço no disco para usar o comando SAVE.

Todos (S/N)?

Quando desejar eliminar todos arquivos de um disco com o comando ERA

MENSAGENS DO HB-DOS

As mensagens enviadas à tela pelo HB-DOS dividem-se em dois grupos: mensagens de apoio a operação do sistema e mensagens de erro.

MENSAGENS DE APOIO A OPERAÇÃO DO SISTEMA

Aperte uma tecla

O HB-DOS mostra esta mensagem após um PAUSE, na execução de um arquivo "BATCH"

Encerra auto comando (S/N)

Aparece quando se tecla CTRL-C durante a execução de um arquivo "BATCH".

Insira disco c/ arquivo batch
Aperte uma tecla

Ocorre quando, durante a execução de um arquivo .BAT, o disco que contém esse arquivo é retirado.

Insira disco no drive d:
e pressione qualquer tecla

Permite ao usuário mudar de disco, quando está trabalhando com um único drive.

Tem certeza(S/N)?

Ocorre quando se tecla DEL*.* . O objetivo é confirmar se o usuário deseja realmente eliminar todos os arquivos do disco.

MENSAGENS DE ERRO

```
Coloque disco com sistema
e aperte uma tecla
```

Ocorre quando o HB-DOS não acha o arquivo HBCCP.COM .

```
Comando inexistente
```

Quando o comando digitado pelo usuário não fizer parte dos comandos internos e nem estiver na forma de arquivo de disco (comandos externos).

```
Conteúdo perdeu-se antes da cópia
```

Comando COPY. Quando, na concatenação de arquivos, o arquivo destino estiver também entre os arquivos origem.

```
Data Inválida
```

Formato de data incorreto ou data fora da faixa permitida.

```
Disco protegido para leitura em d:
Cancela, Repete ou Ignora?
```

Ocorre quando o usuário tenta escrever em disco protegido.

```
Drive inválido
```

Quando o nome do drive não estiver correto, em qualquer especificação de drive.

```
Erro de disco em d:
```

A diferença entre esta mensagem e a mensagem abaixo é que neste caso o erro é irreversível e por isso não adianta o usuário repetir a operação.

```
Erro de disco p/ escrita em d:
Erro de disco p/ leitura em d:
Cancela, Repete ou Ignora?
```

Essas mensagens ocorrem quando houver erro de leitura/escrita no disco. Antes de acusar o erro, o HB-DOS tenta executar o comando três vezes.

```
Erro na criação do arquivo
```

O HB-DOS não conseguiu criar o arquivo. Pode ser problema de falta de espaço no disco ou no diretório.

```
Erro na escrita
```

Ocorreu algum erro físico na escrita no disco.

Erro na troca de nomes

Surge no comando REN, quando já existir arquivo com o nome destino.

Espaço insuficiente no disco

Comando COPY. Não há espaço no disco para copiar arquivo.

Hora Inválida

Formato da hora incorreto ou hora fora da faixa permitida.

Não encontrou arquivo

Alguns comandos utilizam nomes de arquivos como parâmetro. Quando o arquivo origem (usado como parâmetro) não é encontrado, essa mensagem aparece.

Não pode ser copiado nele mesmo

Comando COPY. Arquivos origem e destino com o mesmo nome.

Não pronto p/ leitura em d:
Não pronto p/ escrita em d:
Cancela, Repete ou Ignora?

Essas mensagens ocorrem quando o HB-DOS tenta acessar o drive d:, para leitura (escrita) e a porta do mesmo está aberta, ou não tem disco. O usuário tem a opção de repetir a operação (R), cancelar o comando (C), ou continuar, como se a operação no disco estivesse correta (I).

Não tem relógio

Surge quando se usa o comando TIME e não há relógio no micro.

Parâmetro inválido

Comando MODE. Aparece quando o parâmetro largura da tela for menor ou igual a zero, maior que 40 (caso não haja cartão de 80 colunas) ou maior que 80 (se houver cartão de 80 colunas).

Programa muito longo

É uma mensagem que aparece quando o usuário tenta executar um programa maior que a área disponível de memória.

APÊNDICE IV

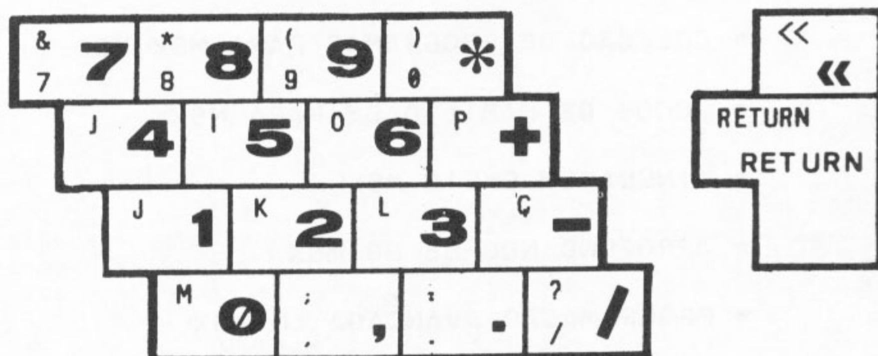
TECLADO NUMÉRICO REDUZIDO DO HB-MCP

O sistema operacional HB-MCP possibilita a utilização de parte do teclado como um "teclado numérico reduzido", facilitando a introdução de dados numéricos.

Para entrar ou sair do modo TECLADO NUMÉRICO deve-se pressionar simultaneamente as teclas: GRAPH e CAPS .

Ao entrar no modo TECLADO NUMÉRICO apenas algumas teclas ou combinações delas continuam operando normalmente, sendo que as demais perdem sua função inicial.

As teclas originais e suas novas configurações obedecem ao quadro abaixo:



Os caracteres no lado esquerdo de cada tecla correspondem ao que realmente está impresso no teclado e, os do lado direito, correspondem aos que estão sendo simulados no modo TECLADO NUMÉRICO.

Os comandos CTRL+STOP (para ressetar o sistema), GRAPH+1 (cópia da tela) e GRAPH+CAPS (para voltar ao teclado normal) continuam ativos!

Enquanto o modo TECLADO NUMÉRICO estiver ativo, o LED (indicador luminoso) da tecla CAPS ficará piscando.

COLEÇÃO MSX

Se você quiser receber gratuitamente dicas de programação e informações sobre outros livros da COLEÇÃO MSX, envie seu nome e endereço completos para:

ALEPH Publicações e
Assessoria Pedagógica Ltda.
Caixa Postal: 20707
CEP: 01498 - São Paulo SP

OUTROS LIVROS DA COLEÇÃO MSX

* COLEÇÃO DE PROGRAMAS PARA MSX VOL.I
* COLEÇÃO DE PROGRAMAS PARA MSX VOL.II
* JOGOS DE HABILIDADE PARA MSX
* LINGUAGEM BASIC MSX
* APROFUNDANDO-SE NO MSX
* PROGRAMAÇÃO AVANÇADA EM MSX
* LINGUAGEM DE MÁQUINA PARA MSX
* HOT LOGO - Primeiros Passos
* HOTPLAN
* HOTWORD
* HOTDATA

COLEÇÃO MSX

USANDO O DISK-DRIVE NO MSX.

RUBENS PEREIRA SILVA JR.

Com o lançamento dos disk drives, os micros MSX tornam-se definitivamente profissionais!
Neste livro são estudados detalhadamente e de forma extremamente didática três "gigantes" da linha MSX: o DSK-BASIC, o HB-MCP e o HB-DOS.
A parte inicial desta obra é dedicada ao leitor que nunca usou um disk drive, explicando o que é e para que serve este periférico, mostrando ainda como usar os disquetes, o que são arquivos, registros, campos, etc...
A segunda parte é sobre o DSK-BASIC, uma extensão do poderoso HOT-BASIC que permite o acesso aos dados gravados em disco diretamente através do BASIC.
A seguir, na terceira parte, o autor discute detalhadamente usos e aplicações de um dos mais poderosos e versáteis sistemas operacionais existentes atualmente: o HB-MCP.
A quarta parte é dedicada ao HB-DOS, o sistema operacional original dos micros MSX.
Complementando a obra o autor, Rubens Pereira Silva Jr., fornece um utilíssimo dicionário de todos os comandos e funções do DSK-BASIC, HB-MCP e HB-DOS para facilitar a consulta rápida.
Trata-se de uma obra de consulta e aprendizado indispensável para conhecer os recursos oferecidos pelo disk drive e operar adequadamente este importantíssimo periférico.

Editora Aleph